TUGAS AKHIR – KS 141501

OPTIMASI PENJADWALAN UJIAN OTOMATIS DENGAN MENGGUNAKAN ALGORITMA *GREEDY – LATE ACCEPTANCE – HYPER HEURISTIC*

PUTRI CAHYANING BWANANESIA
NRP 5210 100 142

Dosen Pembimbing :
Ahmad Muklason, S.Kom., M.Sc., Ph.D.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR – KS 141501

OPTIMASI PENJADWALAN UJIAN OTOMATIS DENGAN MENGGUNAKAN ALGORITMA *GREEDY* - *LATE ACCEPTANCE* – *HYPER HEURISTIC*

PUTRI CAHYANING BWANANESIA
NRP 5210 100 142

Dosen Pembimbing :
Ahmad Muklason, S.Kom., M.Sc., Ph.D.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

Halaman ini sengaja dikosongkan.

FINAL PROJECT – KS 141501

AUTOMATED EXAMINATION TIMETABLING OPTIMIZATION USING *GREEDY* - *LATE ACCEPTANCE* – *HYPER HEURISTIC* ALGORITHM

PUTRI CAHYANING BWANANESIA
NRP 5210 100 142

Supervisor :
Ahmad Muklason, S.Kom., M.Sc., Ph.D.

DEPARTMENT OF INFORMATION SYSTEMS
Faculty of Information Technology and Communication
Sepuluh Nopember Institute of Technology
Surabaya 2018

Halaman ini sengaja dikosongkan.

LEMBAR PERSETUJUAN

OPTIMASI PENJADWALAN UJIAN OTOMATIS DENGAN MENGGUNAKAN ALGORITMA *GREEDY – LATE ACCEPTANCE – HYPER HEURISTIC*

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

PUTRI CAHYANING BWANANESIA
5210 100 142

Disetujui Tim Penguji: Tanggal Ujian . 10 Januari 2018
Periode Wisuda : Maret 2018

Ahmad Muklason, S.Kom., M.Sc., Ph.D. **(Pembimbing I)**

Edwin Riksakomara, S.Kom., M.T. **(Penguji I)**

Wiwik Anggraeni, S.Si., M.Kom. **(Penguji II)**

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

OPTIMASI PENJADWALAN UJIAN OTOMATIS DENGAN MENGGUNAKAN ALGORITMA *GREEDY – LATE ACCEPTANCE – HYPER HEURISTIC*

TUGAS AKHIR

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada

Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

PUTRI CAHYANING BWANANESIA
NRP 5210 100 142

Surabaya, 18 Januari 2018
PLH Kepala
Departemen Sistem Informasi

Edwin Riksakomara, S.Kom., M.T.
NIP. 19690725 200312 1 001

Halaman ini sengaja dikosongkan

OPTIMASI PENJADWALAN UJIAN OTOMATIS DENGAN MENGGUNAKAN ALGORITMA *GREEDY* – *LATE ACCEPTANCE* – *HYPER HEURISTIC*

Nama Mahasiswa : PUTRI CAHYANING B.
NRP : 5210 100 142
Departemen : SISTEM INFORMASI FTIK-ITS
Dosen Pembimbing : Ahmad Muklason, S.Kom., M.Sc., Ph.D

ABSTRAK

Permasalahan mengenai penjadwalan ujian masih menjadi topik yang menarik untuk dipecahkan. Berbagai pendekatan dan metode telah dilakukan untuk mendapat hasil penjadwalan ujian yang optimal. Penjadwalan ujian yang sebelumnya menggunakan penjadwalan manual diharapkan bisa beralih menggunakan solver penjadwalan ujian otomatis sehingga tidak menyita banyak waktu. Selain itu, penjadwalan ujian otomatis diharapkan bisa mendukung mahasiswa memperoleh nilai yang maksimal. Tugas akhir ini bertujuan untuk membuat sebuah solver penjadwalan ujian otomatis yang optimal dengan menggunakan algoritma hyper-heuristic, yang menggabungkan dua algoritma yaitu : algoritma greedy dan algoritma late acceptance.

Metode pengerjaan tugas akhir ini terdiri dari dua fase. Fase pertama, dengan menggunakan algoritma greedy, dibentuk sebuah solusi inisial berupa jadwal ujian yang sudah memenuhi semua hard constraint. Fase kedua, dengan menggunakan algoritma late acceptance, dilakukan optimasi terhadap jadwal ujian yang sudah diperoleh di fase pertama.

Hasil dari algoritma tersebut adalah sebuah jadwal ujian yang optimal dengan nilai proximity cost senilai 36,083. Dengan menggunakan solver penjadwalan ujian otomatis, pembuatan jadwal ujian bisa dilakukan secara lebih efektif dan efisien.

Kata kunci — penjadwalan ujian, penjadwalan otomatis, algoritma greedy, algoritma late acceptance, hyper-heuristic.

AUTOMATED EXAMINATION TIMETABLING OPTIMIZATION USING *GREEDY – LATE ACCEPTANCE – HYPER HEURISTIC* ALGORITHM

Name : PUTRI CAHYANING B.
NRP : 5210 100 142
Department : INFORMATION SYSTEMS FTIK-ITS
Supervisors : Ahmad Muklason, S.Kom., M.Sc.,Ph.D.

ABSTRACT

Examination timetabling problem is still an interesting topic to solve. Various approaches and methods have been made to obtain an optimal exam timetabling results. Exam timetabling that has done manually in previous work is expected to switch to an automatic exam timetabling using solver, so it does not take much time. In addition, automatic exam timetabling is expected to support students get the best exam score. This final project aims to create an optimal automatic exam timetabling solver using hyper-heuristic algorithm, which combine two algorithms : greedy algorithm and late acceptance algorithm.
The method of this final project consist of two phases. In the first phase, the implementation of greedy algorithm formed an initial solution. This initial solution is consider as an exam timetabling that has filled all hard constraints. In the second phase, the implementation of late acceptance algorithm optimize the initial solution which has been obtained in the first phase. The result of this final project is an optimal exam timetabling with a value of proximity cost of 36,083. By using exam timetabling solver, exam timetable making can be done more effectively and efficiently.

Keywords — examination timetabling, automated timetabling, greedy algorithm, late acceptance algorithm, hyper-heuristic

KATA PENGANTAR

Puji syukur atas karunia yang telah diberikan Allah SWT selama ini sehingga penulis mendapatkan kelancaran dalam menyelesaikan tugas akhir dengan judul:

OPTIMASI PENJADWALAN UJIAN OTOMATIS DENGAN MENGGUNAKAN ALGORITMA *GREEDY* – *LATE ACCEPTANCE* – *HYPER HEURISTIC*

Terima kasih atas pihak-pihak yang telah mendukung, memberikan saran, motivasi, semangat, dan bantuan baik materi maupun spiritual demi tercapainya tujuan pembuatan tugas akhir ini. Secara khusus penulis menyampaikan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Bapak Ahmad Mukhlason, S.Kom, M.Sc. selaku dosen pembimbing yang telah memberikan ilmu, bimbingan, dan motivasi untuk kelancaran tugas akhir ini.
2. Bapak Edwin Riksakomara, S.Kom., M.T. dan Ibu Wiwik Anggraeni, S.Si., M.Kom. selaku dosen penguji yang telah memberikan kritik, saran, dan masukan yang dapat menyempurnakan Tugas Akhir ini.
3. Ibu Mahendrawathi ER., ST., M.T. selaku dosen wali penulis selama menempuh pendidikan di Departemen Sistem Informasi.
4. Ibu Feby Artwodini, S.Kom., M.T. selaku sekretaris Departemen Sistem Informasi yang telah memberikan dukungan dan saran tentang penjadwalan ujian.
5. Seluruh dosen pengajar beserta staf Tata Usaha yang telah memberikan ilmu dan bantuan kepada penulis selama menempuh pendidikan di Departemen Sistem Informasi, FTIK ITS Surabaya.

Penyusunan laporan ini masih jauh dari sempurna. Untuk itu, penulis menerima adanya kritik dan saran yang membangun untuk perbaikan di masa mendatang. Semoga buku tugas akhir ini dapat memberikan manfaat pembaca.

Surabaya, Januari 2018

Penulis

Halaman ini sengaja dikosongkan.

DAFTAR ISI

Halaman ini sengaja dikosongkan.

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan.

DAFTAR TABEL

DAFTAR KODE PROGRAM

Halaman ini sengaja dikosongkan.

BAB I
PENDAHULUAN

Pada bab ini, akan dijelaskan tentang Latar Belakang Masalah, Perumusan Masalah, Batasan Masalah, Tujuan Tugas Akhir, Manfaat Kegiatan Tugas Akhir dan Relevansi dengan laboratorium RDIB.

1.1. Latar Belakang

Ujian menjadi salah satu faktor penentu keberhasilan studi mahasiswa. Dalam satu semester, setidaknya ada dua ujian wajib yang harus diikuti oleh mahasiswa, yaitu Ujian Tengah Semester (UTS) dan Ujian Akhir Semester (UAS). Agar mahasiswa bisa mencapai hasil akhir yang memuaskan, maka pihak Departemen sebagai penyelenggara ujian harus menyiapkan support system yang baik. Salah satunya adalah dengan membuat jadwal ujian yang optimal bagi mahasiswa, sehingga mahasiswa tidak merasa dirugikan dan bisa mengatur waktu belajar untuk mempersiapkan ujiannya dengan lebih maksimal.

Namun sayangnya, penjadwalan ujian masih menjadi suatu permasalahan yang menyita waktu. Pihak Departemen sebagai penyelenggara ujian masih menggunakan metode manual dalam membuat jadwal ujian. Pembuatan jadwal ujian manual ini bisa memakan waktu sampai 1-2 minggu. Selain itu, jadwal ujian yang manual ini sangat tidak fleksibel, sehingga akan menyulitkan ketika ada perubahan mendadak. Masalah lain yang mungkin muncul adalah dari pihak mahasiswa, masih ada beberapa mahasiswa yang mendapat jadwal ujian yang bersamaan untuk mata kuliah yang berbeda. Hal ini tentu saja tidak memungkinkan bagi mahasiswa tersebut untuk mengikuti dua ujian sekaligus dalam satu waktu yang sama. Jadwal yang manual ini juga dianggap belum optimal karena dalam sehari mahasiswa bisa mengikuti dua ujian atau lebih. Padahal seharusnya akan lebih baik jika mahasiswa hanya mengikuti

satu ujian saja dalam sehari, sehingga mahasiswa tersebut bisa lebih fokus menyiapkan ujiannya dan mendapat hasil yang baik.

1.2. Perumusan Masalah

Berdasarkan uraian latar belakang yang telah dijelaskan, maka rumusan masalah dari tugas akhir ini, yaitu :

a. Bagaimana membuat model matematis untuk permasalahan penjadwalan ujian di Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya?
b. Bagaimana membuat *solver* penjadwalan ujian otomatis yang dapat memenuhi semua aturan dan batasan yang ditentukan oleh Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya?
c. Bagaimana membuat *solver* penjadwalan ujian otomatis yang dapat mengoptimalkan jadwal ujian mahasiswa?
d. Bagaimana menerapkan algoritma *greedy – late acceptance – hyper heuristic* untuk menyelesaikan permasalahan penjadwalan ujian?
e. Bagaimana perbandingan hasil penjadwalan ujian secara manual dan penjadwalan ujian otomatis oleh *solver* penjadwalan ujian?

1.3. Batasan Pengerjaan Tugas Akhir

Batasan pemasalahan dalam tugas akhir ini, yaitu :

a. Data yang digunakan dalam tugas akhir ini yaitu data peserta dan data mata kuliah yang diujikan pada UTS & UAS semester genap tahun 2016 – 2017 di Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya.
b. Hasil dari tugas akhir ini adalah sebuah *solver* penjadwalan ujian otomatis yang dibangun dengan bahasa pemrograman Java dan berbasis lokal.
c. *Solver* penjadwalan ujian otomatis tersebut akan digunakan untuk membuat jadwal UTS & UAS di Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

d. Fokus pada penjadwalan berbasis waktu ujian, tanpa menyebutkan lokasi ruang ujian.
e. Kapasitas total ruang ujian digunakan sebagai *hard constraints.*
f. *Tools* yang digunakan untuk pembuatan *solver* penjadwalan ujian otomatis adalah Netbeans IDE.

1.4. Tujuan Tugas Akhir

Tujuan yang hendak dicapai dalam pengerjaan tugas akhir ini adalah membuat *solver* penjadwalan ujian otomatis yang optimal untuk penjadwalan UTS & UAS di Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya dengan menggunakan algoritma *greedy – late acceptance* – hyper-heuristic. *Solver* penjadwalan ujian otomatis tersebut diharapkan bisa menggantikan sistem manual yang digunakan sebelumnya dan menghasilkan jadwal ujian yang optimal bagi mahasiswa.

1.5. Manfaat Tugas Akhir

Manfaat yang diberikan dengan adanya tugas akhir ini adalah sebagai berikut:

1. Manfaat Teoritis

Adanya tugas akhir ini diharapkan dapat menambah khasanah keilmuan dalam bidang sistem informasi khususnya mengenai implementasi algoritma *greedy – late acceptance – hyper heuristic* untuk membuat *solver* penjadwalan ujian otomatis.

2. Manfaat Praktis

a. Bagi Pihak Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember Surabaya sebagai Klien

Tugas akhir ini diharapkan dapat membantu Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya sebagai penyelenggara ujian untuk membuat jadwal ujian secara otomatis sehingga dapat menghemat waktu dan dapat menghasilkan jadwal ujian yang optimal bagi mahasiswa.

b. Bagi Mahasiswa

Tugas akhir ini diharapkan dapat membantu mahasiswa memperoleh nilai yang maksimal dalam ujian. Mahasiswa dapat memperoleh jadwal ujian yang optimal, tidak ada mahasiswa yang mempunyai jadwal yang bertabrakan (harus mengikuti dua ujian dalam waktu yang bersamaan), juga dalam satu hari mahasiswa diharapkan dapat mengikuti satu ujian saja sehingga lebih fokus belajar untuk mempersiapkan ujian tersebut.

c. Bagi Penelitian Berikutnya

Tugas akhir ini diharapkan dapat memberikan motivasi bagi peneliti lain untuk melakukan pengembangan terhadap studi mengenai penjadwalan ujian dengan menggunakan algoritma atau pendekatan lainnya.

1.6. Relevansi

Topik tugas akhir ini yaitu mengenai optimasi penjadwalan ujian. Topik tersebut sesuai dengan bidang ilmu riset operasi yang menjadi cakupan research roadmap pada laboratorium Rekayasa Data dan Inteligensi Bisnis (RDIB).

BAB II
TINJAUAN PUSTAKA

Sebelum melakukan pengerjaan tugas akhir, penulis melakukan tinjauan pustaka dari beberapa penelitian sebelumnya dan dasar teori yang sesuai dengan topik tugas akhir.

2.1. Penelitian Terdahulu

Pada sub bab ini akan diterangkan mengenai beberapa penelitian terdahulu yang telah dilakukan dan memiliki relevansi dengan tugas akhir ini. Penelitian terdahulu tersebut dapat dilihat pada tabel 5.1, 5.2, dan 5.3.

Tabel 2.1 Penelitian Terdahulu (1)

Nama Peneliti	Michael W. Carter, Gilbert Laporte dan Sau Yan Lee [1]
Tahun Penelitian	1996
Judul Penelitian	*Examination Timetabling : Algorithmic Strategies and Applications*
Penjelasan Singkat	Penjadwalan ujian adalah sebuah permasalahan operasional yang dihadapi oleh banyak institusi pendidikan. Peneliti telah membuat sebuah sistem penjadwalan ujian terkomputerisasi yang disebut dengan *EXAMINE*. Pada penelitian ini, peneliti menggunakan dan membandingkan beberapa strategi algoritma untuk menyelesaikan permasalahan penjadwalan ujian. EXAMINE dan beberapa algoritma penjadwalan ujian lainnya memasukkan aspek biaya untuk menilai kualitas dari jadwal potensial. Setiap institusi pendidikan pasti mempunyai permasalahan ujian yang berbeda. Contohnya, di Universitas Toronto, masalah yang dihadapi adalah menjadwalkan ujian dengan kapasitas ruangan

	yang terbatas. Sedangkan permasalahan yang dihadapi di Carleton adalah ujian yang harus dilaksanakan pada saat sore hari. Hal inilah yang menyulitkan para peneliti untuk membandingkan algoritma-algoritma penjadwalan ujian, atau untuk membandingkan permasalahan penjadwalan ujian yang berbeda. Penelitian ini mencoba menyelesaikan kedua isu tersebut dan berusaha membuat sebuah solusi bebas konflik dengan menghasilkan jadwal ujian yang tersebar / terdistribusi dengan baik.
Hasil Penelitian	Pada penelitian ini, satu-satunya *hard constraint* yang harus dipenuhi adalah konflik ujian, artinya dalam dua ujian yang mempunyai mahasiswa yang sama tidak boleh berjalan dalam satu periode waktu yang sama. Sebagai tambahan, terdapat *soft constraint* berupa rentang periode ujian, dimana dua ujian yang mempunyai mahasiswa yang sama harus dijadwalkan dalam beberapa periode terpisah. Oleh karena itu, tujuan penjadwalan ujian dalam penelitian ini adalah meminimalkan jumlah *proximity cost* per mahasiswa dengan membuat penyebaran ujian yang seluas mungkin untuk tiap mahasiswa. Artinya, jika seorang mahasiswa mengikuti satu ujian dan langsung melanjutkan ujian lain pada periode berikutnya, maka penalti yang diberikan akan lebih besar dibanding jika mahasiswa tersebut mempunyai waktu istirahat sebelum mengikuti ujian selanjutnya. Jika mahasiswa mempunyai waktu istirahat yang lebih panjang dari 5 periode, maka penaltinya bernilai nol.

Tabel 2.2 Penelitian Terdahulu (2)

Nama Peneliti	Ahmad Muklason [2]
Tahun Penelitian	2017
Judul Penelitian	*Solver* Penjadwal Ujian Otomatis Dengan Algoritma *Maximal Clique* dan *Hyper-heuristics*
Penjelasan Singkat	Penelitian ini membahas usulan metode baru yaitu metode *heuristic* sekuensial berdasarkan konsep *maximal clique* pada teori graf digabung dengan metode *hyper-heuristic*. Secara umum, algoritma yang digunakan pada penelitian ini terdiri dari dua fase. Fase pertama menggunakan algoritma *sequential heuristic* berdasarkan *maximal clique* yang bertujuan untuk mengonstruksi solusi inisial yang *feasible,* yaitu memenuhi semua *hard constraints.* Sedangkan fase kedua menggunakan algoritma *hyper-heuristics* yang bertujuan untuk mengoptimalkan solusi inisial, yaitu meminimalkan jumlah penalti akibat pelanggaran terhadap *soft constraints.*
Hasil Penelitian	Kontribusi utama dari penelitian ini adalah algoritma baru dalam *framework hyper-heuristic* yang merupakan gabungan dari beberapa algoritma, yaitu : *maximal clique – saturation degree based sequential heuristic* dan *self adaptive learning – great deluge hyper-heuristic.* Hasil penelitian komputasi menunjukkan bahwa metode ini sangat efektif untuk memecahkan permasalahan penjadwalan ujan dan lebih unggul jika dibandingkan dengan hasil penelitian sebelumnya dengan metode lain.

Tabel 2.3 Penelitian Terdahulu (3)

Nama Peneliti	Ahmad Muklason, Andrew J. Parkes, Ender Özcan, Barry McCollum, Paul McMullan [3]
Tahun Penelitian	2017
Judul Penelitian	*Fairness in examination timetabling: Student preferences and extended formulations*
Penjelasan Singkat	Penelitian ini menampilkan hasil survey mengenai preferensi mahasiswa sehubungan dengan jadwal ujian mereka, yang bertujuan untuk memperoleh solusi yang dapat memuaskan preferensi mahasiswa dan semua pertimbangan *benchmark* yang ada. Salah satu fokus utama dalam penelitian ini berkaitan dengan aspek keadilan atau perlakuan yang sama bagi setiap mahasiswa, contohnya : mahasiswa mempertimbangkan keadilan terhadap jadwal ujiannya dan jadwal ujian temannya. Cara mengukur aspek keadilan tersebut adalah dengan memberikan alokasi waktu yang adil bagi setiap mahasiswa untuk mempersiapkan diri sebelum ujian. Aspek keadilan ini diperkenalkan sebagai salah satu tujuan baru sebagai tambahan untuk tujuan-tujuan standar yang sudah ada sebelumnya, sehingga menciptakan sebuah permasalahan *multi-objective*. Data yang digunakan dalam penelitian ini adalah beberapa model data ujian sebenarnya yang kemudian diperluas dan setiap *benchmark* digunakan dalam penelitian untuk mendefinisikan efektivitas dari sebuah pendekatan *multi-stage multi-objective* berdasarkan skalarisasi *weighted* Tchebyceff yang bertujuan meningkatkan aspek keadilan dan mencapai tujuan-tujuan lainnya.

Hasil Penelitian	Survey yang dilakukan terhadap para mahasiswa menunjukkan bahwa setengah dari jumlah mahasiswa merasa tidak puas dengan penjadwalan ujian yang mereka dapatkan. Selain itu, sekitar 30% responden mahasiswa percaya bahwa penjadwalan ujian tersebut berdampak negatif pada pencapaian akademik mereka. Penelitian ini berusaha memenuhi aspek keadilan bagi setiap mahasiswa dengan menggunakan algoritma *hyper-heuristics* yang digabungkan dengan optimasi *multi-objective*. Hasilnya, model dan metode yang didapatkan dalam eksperimen dapat membuat solusi jadwal ujian yang berkualitas tinggi. Selain itu juga memunculkan *trade-off* antara *soft constraints* standar dan meningkatkan aspek keadilan bagi mahasiswa.

2.2. Penjadwalan Ujian

Penjadwalan merupakan sebuah permasalahan menarik yang telah diteliti selama lebih dari empat dekade di berbagai bidang ilmu, khususnya di bidang riset operasi dan kecerdasan buatan. Penjadwalan adalah permasalahan optimasi kombinatorik yang didefinisikan oleh By Lawler sebagai :

"Studi matematis untuk mencari solusi optimal pada penyusunan, pengelompokan, pengurutan atau pemilihan objek diskrit yang biasanya berjumlah terbatas *(finite number)*." [4]

Secara khusus, menurut Burke, penjadwalan mempunyai definisi sebagai berikut.

" Permasalahan penjadwalan adalah permasalahan yang mempunyai empat parameter, yaitu T (himpunan dari *times* / alokasi waktu), R (himpunan dari *resources* / sumber daya), M (himpunan dari *meetings* / pertemuan), dan C (himpunan dari *constraints* / batasan). Penjadwalan digunakan untuk

mengalokasikan waktu dan sumber daya pada pertemuan tertentu dengan memenuhi batasan paling maksimal." [5]

Sedangkan menurut Schaerf, permasalahan tentang penjadwalan dapat dibagi menjadi beberapa bagian, yaitu : permasalahan penjadwalan sekolah, permasalahan penjadwalan per mata pelajaran / per mata kuliah, dan permasalahan penjadwalan ujian. Schaerf juga menyatakan bahwa tidak ada perbedaan yang mencolok diantara ketiganya. Sebuah permasalahan penjadwalan bisa mencakup salah satu atau diantara dua bagian yang telah dijelaskan tersebut. [6]

Permasalahan penjadwalan ujian juga bisa dilihat dari sudut pandang yang berbeda. Contohnya, permasalahan penjadwalan ujian dapat dinyatakan sebagai permasalahan alokasi sumber daya, permasalahan pencarian dan optimasi, juga dapat pula dinyatakan sebagai permasalahan batasan kepuasan. Jika melihat dari sudut pandang permasalahan batasan kepuasan serta permasalahan pencarian dan optimasi, Burke merepresentasikan penjadwalan sebagai 3-tupel $\langle E, I, K \rangle$ yang menyatakan E sebagai himpunan *examination* / ujian, I sebagai himpunan *resource* / sumber daya, contohnya alokasi waktu, dan K sebagai himpunan *constraints* / batasan. Secara matematis ditulis sebagai berikut.

$E = \{e_1, e_2, e_3, \dots, e_N\}$ adalah himpunan ujian per mata kuliah

$K = \{k_1, k_2, k_3, \dots, k_M\}$ adalah himpunan batasan

$T = \{t_1, t_2, t_3, \dots, t_L\}$ adalah himpunan alokasi waktu

Sehingga permasalahan penjadwalan ujian dapat didefinisikan sebagai pencarian pengalokasian terbaik untuk $\forall_x, \exists_y (e_x = t_y)$ yang artinya ujian e_x dijadwalkan pada alokasi waktu t_y dimana $e_x \in E$ dan $t_y \in T$ sedemikian hingga semua batasan K terpenuhi. [7]

Tujuan umum yang ingin dicapai dalam permasalahan penjadwalan adalah memenuhi semua *hard constraints* dan meminimalisasi pelanggaran terhadap *soft constraints*, yang dapat bervariasi tergantung dari kebijakan masing-masing instansi. Batasan-batasan ini telah diteliti oleh Burke [8]. Solusi yang dapat memenuhi semua *hard constraints* disebut sebagai solusi yang mungkin *(feasible solution)*. Apabila solusi tersebut

juga dapat memenuhi semua *soft constraints* (tidak ada pelanggaran apapun terhadap *soft constraints*), maka solusi itu disebut sebagai solusi yang sempurna *(perfect solution)*. Akan tetapi, untuk menghasilkan sebuah solusi yang optimal, adanya solusi yang sempurna menjadi tidak begitu penting. Contohnya dalam hal penjadwalan ujian, mungkin saja terjadi pelanggaran terhadap salah satu *soft constraints* yaitu jarak antara dua ujian mata kuliah setidaknya 5 *timeslots*.

Menurut teori kompleksitas komputasi, permasalahan penjadwalan ujian diklasifikasikan sebagai NP-*complete problem* [9]. Dengan demikian, belum ada algoritma eksak yang mampu menemukan solusi optimal dalam waktu polinomial. Oleh karena itulah, permasalahan ini masih menjadi permasalahan yang menarik untuk dikaji. Pada beberapa literatur juga disebutkan beberapa *benchmark dataset* untuk permasalahan penjadwalan ujian ini. *Dataset* yang umum digunakan adalah *Carter dataset* dan *International Timetabling Competition 2007 dataset* [10].

Pada tugas akhir ini, *dataset* yang akan digunakan adalah data peserta dan data mata kuliah yang diujikan pada UTS & UAS semester genap tahun ajaran 2016 – 2017 di Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Penjelasan mengenai *dataset* ini akan dijabarkan pada sub bab 5.2.2.

2.3. Dataset UTS & UAS

Data yang digunakan pada tugas akhir ini adalah data peserta dan data mata kuliah yang diujikan pada Ujian Tengah Semester (UTS) dan Ujian Akhir Semester (UAS) semester genap tahun ajaran 2016 – 2017 pada Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Data yang didapatkan antara lain : daftar mata kuliah yang diujikan, daftar mahasiswa yang mengikuti ujian, dan daftar ruang ujian. Selain data mata kuliah yang diujikan, terdapat pula data mahasiswa yang mengikuti ujian, jadwal ujian (hari dan & sesi ujian), serta ruang ujiannya. Tabel 5.4 menunjukkan ringkasan dari data yang didapatkan.

Tabel 2.4. Ringkasan Data

Ujian	Jumlah Mata Kuliah	Jumlah Mahasiswa	*Time Slot*	Jumlah Ruang Ujian
UTS	32	567	15	10
UAS	32	519	15	10

Penjadwalan ujian tersebut merupakan jadwal ujian yang dibuat secara manual. Jadwal inilah yang nantinya akan digunakan sebagai pembanding jadwal ujian otomatis yang merupakan hasil / keluaran dari tugas akhir ini.

Data mentah yang telah didapatkan dari pihak Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya perlu melalui mekanisme pra proses terlebih dahulu agar menghasilkan data yang berkualitas baik dan siap digunakan untuk proses selanjutnya. Tugas akhir ini menggunakan data *benchmark* dari Universitas Toronto atau data yang biasa disebut dengan *Carter Dataset* [11], sehingga data yang didapat akan disesuaikan dengan format *Carter Dataset*.

2.4. Algoritma *Greedy*

Algoritma *greedy* merupakan algoritma yang paling populer digunakan untuk memecahkan permasalahan optimasi. Optimasi bertujuan untuk mencari solusi optimum bagi suatu masalah. Hanya ada dua macam persoalan optimasi, yaitu maksimasi dan minimasi. Dalam algoritma *greedy*, permasalahan optimasi diselesaikan secara bertahap. Prinsip yang digunakan pada algoritma *greedy* adalah mengambil keuntungan sebanyak mungkin yang bisa diambil pada saat itu, tanpa memperhatikan konsekuensi yang akan muncul selanjutnya. Oleh karena itu, pada setiap tahapnya dipilih keputusan yang terbaik pada saat itu saja. Itulah yang selanjutnya disebut dengan optimum lokal *(local optimum)*, dengan harapan bahwa pada setiap tahapan tersebut dapat mengarah ke solusi optimum global *(global optimum)* [12]. Terdapat lima elemen dalam algoritma *greedy*, antara lain :

1. Himpunan kandidat (C) : himpunan yang berisi calon-calon elemen yang dapat membentuk solusi.
2. Himpunan solusi (S) : himpunan yang berisi elemen-elemen yang merupakan solusi dari permasalahan.
3. Fungsi seleksi : fungsi yang digunakan untuk memilih langkah terbaik pada saat itu, yaitu dengan memilih optimum lokal untuk mencapai optimum global.
4. Fungsi kelayakan : fungsi yang digunakan untuk menentukan apakah sebuah langkah memenuhi syarat dari permasalahan. Dengan kata lain, langkah yang dipilih tersebut tidak boleh melanggar batasan-batasan yang ditetapkan.
5. Fungsi obyektif : fungsi yang bertujuan memaksimalkan atau meminimalkan solusi yang diperoleh [13].

Dengan kata lain, algoritma *greedy* melibatkan pencarian sebuah himpunan bagian S dari himpunan kandidat C, dengan syarat S harus memenuhi beberapa kriteria yang ditentukan, yaitu menyatakan suatu solusi dan S dioptimasi oleh sebuah fungsi obyektif.

Algoritma *greedy* juga mempunyai kelemahan, yaitu solusi optimum global yang dihasilkan belum tentu menjadi solusi optimum yang terbaik, tetapi sub-optimum atau *pseudo-optimum*. Hal ini bisa terjadi karena algoritma *greedy* tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada. Selain itu, terdapat pilihan beberapa fungsi seleksi yang berbeda, sehingga harus dipilih fungsi seleksi yang tepat untuk menghasilkan solusi optimal. Oleh karena itulah, penggunaan algoritma *greedy* tidak selalu berhasil memberikan solusi optimal sehingga akan lebih baik jika algoritma ini digabungkan dengan algoritma lain untuk mendapatkan solusi optimal.

2.5. Algoritma *Late acceptance*

Late acceptance adalah sebuah pendekatan baru mengenai teknik *meta-heuristic* dengan tujuan umum. Algoritma ini diusulkan oleh Burke dan Bykov dalam penelitiannya mengenai permasalahan penjadwalan ujian. Meskipun saat ini penelitian

mengenai penggunaan algoritma *late acceptance* masih dalam tahap awal, metode ini terbukti telah menunjukkan performa yang menjanjikan.

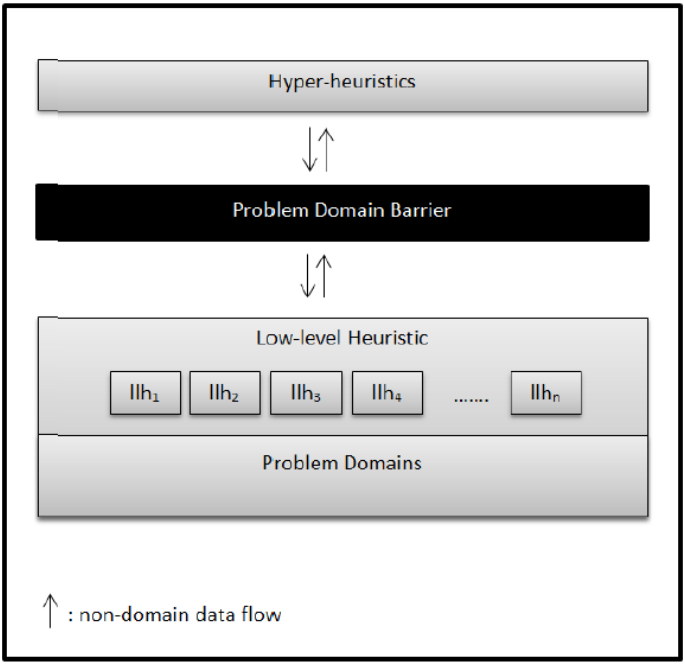
Algoritma *late acceptance* termasuk dalam kelompok teknik pencarian iteratif tetapi menggunakan mekanisme penerimaan yang lebih canggih. Hal ini tidak lazim digunakan pada sebagian besar metode pencarian *meta-heuristic* (contohnya pada *Hill-Climbing* dan *Late acceptance*), dimana pada masing-masing iterasi, kandidat solusi baru yang dihasilkan akan dibandingkan dengan solusi yang sudah ada saat ini. Namun sebaliknya, pada algoritma *late acceptance,* ide utamanya adalah membandingkan kandidat solusi dengan solusi "saat ini" pada beberapa iterasi sebelumnya. Sejalan dengan itu, setiap solusi saat ini tidak langsung digunakan sebagai perbandingan pada saat itu juga, tetapi digunakan pada beberapa iterasi mendatang. "Keterlambatan" dalam perbandingan inilah yang kemudian menginspirasi lahirnya sebuah metode *meta-heuristic* baru, yang kemudian juga memungkinkan penggunaan aturan *greedy acceptance* yang paling sederhana. Versi terakhir dari algoritma *late acceptance* yaitu membandingkan fungsi biaya dari kandidat dengan kompetitor "sebelumnya" dan hanya kandidat dengan biaya yang lebih baik (atau sama) yang diterima. [14]

2.6. Algoritma *Hyper-Heuristic*

Ide dasar dari algoritma *hyper-heuristic* adalah gagasan untuk mengembangkan sebuah algoritma yang lebih umum yang bertentangan dengan pendekatan *meta-heuristic* standar. Umumnya, kelemahan dari algoritma *meta-heuristic* adalah algoritma memerlukan *parameter tuning* yang intensif dan memerlukan pengetahuan *problem domain* yang spesifik. Sehingga, untuk *problem instance* yang berbeda diperlukan *parameter tuning* yang berbeda pula. Hal ini pada akhirnya akan berpengaruh pada performa algoritma. Artinya, satu *problem instance* dapat mempunyai performa algoritma yang baik namun *problem instance* lainnya dapat mempunyai performa algoritma yang sangat buruk. Oleh karena itulah,

dilakukan pendekatan dengan menggunakan algoritma *hyper-heuristic* [2].

Hyper-heuristic didefinisikan sebagai kumpulan pendekatan yang bertujuan untuk mengotomasi proses. Otomasi yang dilakukan tersebut biasanya dilakukan melalui metode *machine learning*. Tujuannya adalah memilih dan mengombinasikan *heuristic* yang sederhana atau menghasilkan *heuristic* baru dengan komponen *heuristic* yang sudah ada, sehingga dapat menyelesaikan permasalahan pencarian komputasi yang sangat sulit dilakukan secara manual [10].



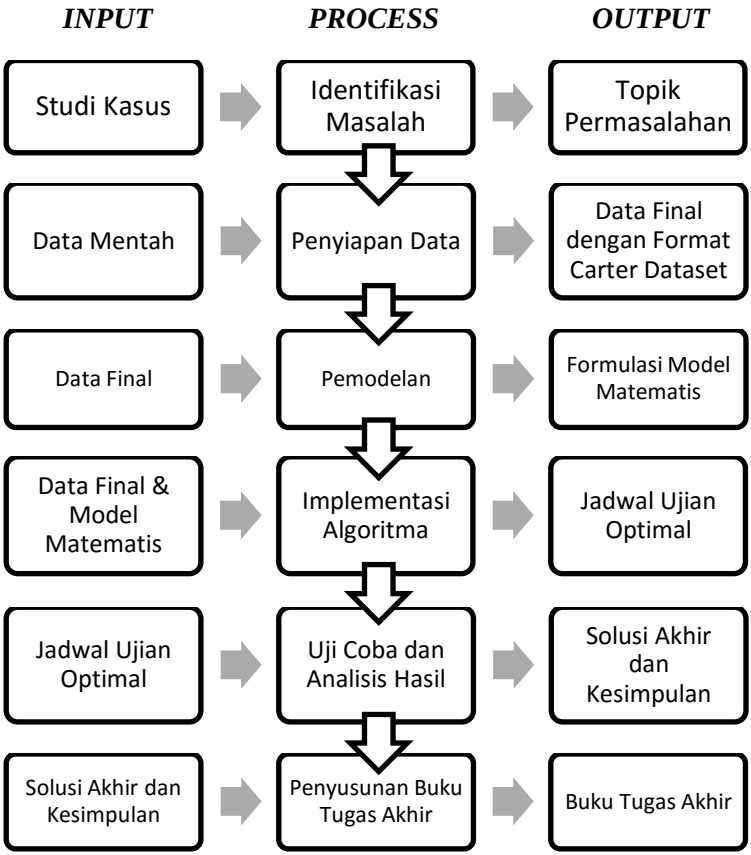
Gambar 2.1 Framework Hyper-heuristic

Gambar 5.1 menunjukkan *framework* dari *hyper-heuristic* [15]. Pada *framework* ini digambarkan bahwa algoritma *hyper-heuristic* menggunakan *problem domain barrier* yang mengakibatkan strategi *hyper-heuristic* tidak bersinggungan

langsung dengan *solution space*, sebagaimana yang biasa terjadi pada pendekatan *meta-heuristic.* Sebaliknya, algoritma *hyper-heuristic* hanya bersinggungan dengan *heuristic* pada level yang lebih rendah. Algoritma *hyper-heuristic* bekerja di atas *search space* berupa *heuristic* pada level yang lebih rendah, sedangkan *meta-heuristic* bekerja di atas *search space* berupa *solution space.* Dengan adanya *search space* yang lebih tinggi ini, algoritma *hyper-heuristic* tidak bergantung pada *problem domain* tertentu saja sehingga tidak diperlukan *parameter tuning* secara manual untuk setiap *problem domain*. Hal ini dikarenakan algoritma *hyper-heuristic* sudah mampu melakukan mekanisme *parameter tuning* secara otomatis [2].

BAB III
METODOLOGI

Pada bab ini akan dijelaskan mengenai alur metodologi yang akan dilakukan dalam tugas akhir ini. Metodologi ini juga digunakan sebagai pedoman untuk melaksanakan tugas akhir agar terarah dan sistematis. Gambar 3.1 menunjukkan metodologi tugas akhir yang digunakan.



Gambar 3.1. Metode Pengerjaan Tugas Akhir

3.1. Identifikasi Masalah

Tahap persiapan adalah tahap untuk memahami permasalahan yang terjadi di Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya mengenai penjadwalan ujian, menetapkan tujuan, dan menentukan batasan dari permasalahan yang akan diselesaikan melalui tugas akhir yang dikerjakan. Selain itu, pada tahap ini juga dilakukan studi literatur mengenai penelitian-penelitian yang telah dilakukan sebelumnya serta studi mengenai pendekatan-pendekatan yang dilakukan untuk memecahkan permasalahan penjadwalan ujian. Studi literatur tersebut berguna sebagai dasar teori yang digunakan sebagai referensi / acuan dalam pengerjaan tugas akhir. Referensi utama yang digunakan pada tugas akhir ini adalah PhD Thesis dari Ahmad Muklason yang berjudul *Hyper-heuristics and Fairness in Examination Timetabling Problems.* Tesis tersebut ditulis pada tahun 2017 sebagai syarat untuk mendapatkan gelar PhD dari Universitas Nottingham. Literatur lain yang digunakan dalam tugas akhir ini yaitu literatur mengenai permasalahan penjadwalan ujian, algoritma *greedy*, algoritma *late acceptance*, dan algoritma *hyper-heuristic.*

3.2. Penyiapan Data

Pada tahap penyiapan data, dilakukan pengumpulan data-data awal yang akan digunakan dalam pengerjaan tugas akhir. Data yang digunakan pada tahap ini adalah data peserta dan data mata kuliah yang diujikan pada Ujian Tengah Semester (UTS) dan Ujian Akhir Semester (UAS) pada semester genap tahun ajaran 2016 – 2017 yang digunakan di Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember Surabaya. Sebagai batasan, data yang diambil hanya data ujian untuk jenjang sarjana (S1). Tahap ini merupakan tahap untuk memahami data yang diperoleh, yaitu memahami tipe datanya, variabel-variabel yang melekat pada data, dan jumlah data. Selain itu, dilakukan pula evaluasi terhadap kualitas data. Tahap selanjutnya adalah

mempersiapkan data final yang akan digunakan untuk proses pemodelan. Data final yang dimaksud adalah data yang telah melalui mekanisme pra proses data sehingga menjadi data yang berkualitas baik dan dapat diolah. Bila perlu, dapat dilakukan proses transformasi pada variabel tertentu untuk menunjang proses pemodelan. Format data yang digunakan akan disesuaikan dengan *Carter Dataset*, dimana data yang semula berbentuk file *excel* akan diubah menjadi file dengan ekstensi .txt. Keluaran dari tahap ini adalah sebuah *dataset* final yang akan digunakan pada tahap berikutnya.

3.3. Pemodelan

Tahap pemodelan adalah tahap pembuatan model matematis dari permasalahan penjadwalan ujian yang ada di Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Pembuatan model tersebut berdasarkan beberapa variabel, yaitu :

1. Variabel Keputusan
   Variabel keputusan berfungsi untuk memastikan bahwa setiap mata kuliah ditempatkan pada satu sesi / slot ujian tertentu.
2. Fungsi Tujuan
   Fungsi tujuan yang digunakan adalah *proximity cost.* Fungsi tujuan dianggap berhasil jika nilai *proximity cost* yang dihasilkan semakin kecil (minimum).
3. Fungsi Batasan
   Jadwal ujian yang dihasilkan harus memenuhi tiga batasan *(hard constraint)*. Ketiga *hard constraint* tersebut antara lain : semua mata kuliah harus mempunyai jadwal ujian, tidak ada jadwal ujian yang bentrok, dan jumlah peserta ujian tidak boleh melebihi kapasitas total ruang ujian.

3.4. Implementasi Algoritma

Pada tahap implementasi, dilakukan pembuatan *solver* penjadwalan ujian otomatis dengan menggunakan *tools* Netbeans IDE 8.2. *Solver* penjadwalan ujian otomatis tersebut

dibuat dengan bahasa pemrograman java dan berbasis lokal. Tahap ini terdiri dari dua fase, yaitu fase implementasi algoritma *greedy* dan fase implementasi algoritma *late acceptance*.

3.4.1. Implementasi Algoritma *Greedy*

Pada fase implementasi yang pertama, dilakukan penerapan algoritma *greedy* dengan menggunakan data final yang telah disesuaikan dengan format *Carter Dataset*. Pada fase ini, data mata kuliah diurutkan berdasarkan jumlah mahasiswa peserta ujian. Lalu setiap mata kuliah dimasukkan ke dalam slot ujian tertentu. Penempatan slot untuk mata kuliah tersebut harus memenuhi dua *hard constraint*, yaitu tidak ada jadwal ujian yang bertabrakan dalam satu slot waktu yang sama serta kapasitas peserta ujian tidak boleh melebihi kapasitas total ruang ujian pada slot waktu tersebut. Apabila suatu mata kuliah memenuhi kedua *hard constraint* tersebut, maka algoritma akan beralih ke penjadwalan untuk mata kuliah berikutnya, namun apabila tidak memenuhi kedua *hard constraint*, maka algoritma akan memindahkan mata kuliah tersebut ke slot lain dan kembali akan memastikan agar kedua *hard constraints* telah terpenuhi. Mata kuliah yang telah memenuhi dua *hard constraint* akan menjadi solusi inisial yang kemudian akan dioptimalkan kembali menggunakan algoritma *late acceptance*. Solusi inisial yang dihasilkan tersebut berupa jadwal ujian yang belum dioptimasi.

3.4.2. Implementasi Algoritma *Late Acceptance*

Pada fase implementasi yang kedua, dilakukan penerapan algoritma *late acceptance* dengan menggunakan solusi inisial yang didapatkan dari fase pertama. Penerapan algoritma *late acceptance* membutuhkan beberapa parameter yang perlu ditentukan terlebih dahulu, seperti parameter jumlah antrian dan jumlah iterasi. Pada fase kedua, jadwal ujian yang menjadi solusi inisial hasil penerapan algoritma *greedy* akan dioptimasi dengan cara mengubah slot ujian untuk mata kuliah tertentu. Perubahan slot ujian dilakukan dengan menukar dua slot ujian

secara acak atau dengan mengganti salah satu slot ujian dengan slot ujian lain secara acak untuk menemukan solusi baru yang menghasilkan jadwal ujian yang lebih optimal. Hasil dari fase ini adalah jadwal ujian yang sudah dioptimalkan dengan meminimalkan nilai *proximity cost.*

3.5. Uji Coba dan Analisis Hasil

Pada tahap uji coba, dilakukan uji performa model dengan melakukan evaluasi terhadap hasil pemodelan yang dihasilkan oleh *solver* penjadwalan ujian. Uji coba dilakukan dengan mengganti nilai parameter untuk mengetahui pengaruh parameter tersebut terhadap nilai *proximity cost* dan jadwal ujian yang dihasilkan.

Selanjutnya, dilakukan analisis terhadap performa algoritma. Pada tahap ini dilakukan perbandingan hasil uji coba dengan menggunakan beberapa parameter yang berbeda. Analisis tersebut bertujuan untuk mengetahui paramater mana yang menghasilkan jadwal ujian yang paling baik, dilihat dari nilai *proximity cost* yang paling minimum.

Selain itu, dilakukan pula analisis terhadap jadwal ujian yang dihasilkan. Pada tahap ini ini, dilakukan perbandingan jadwal ujian. Pertama, membandingkan jadwal ujian yang dibuat secara manual dengan jadwal ujian yang dihasilkan dari penerapan algoritma *greedy*. Kedua, membandingkan jadwal ujian yang dihasilkan dari penerapan algoritma *greedy* dan jadwal ujian yang dihasilkan dari penerapan algoritma *late acceptance.* Ketiga, membandingkan jadwal ujian yang dibuat secara manual dengan jadwal ujian yang dihasilkan dari penerapan algoritma *late acceptance.* Dari hasil analisis jadwal ujian tersebut, dapat diketahui jadwal ujian mana yang berhasil memenuhi semua *hard constraint* yang telah ditetapkan, serta jadwal ujian mana yang memiliki nilai *proximity cost* paling minimum. Jadwal ujian yang memiliki nilai *proximity cost* paling minimum adalah jadwal ujian yang dianggap paling optimal. Jadwal ujian tersebut dapat digunakan sebagai

referensi untuk pembuatan jadwal UTS dan UAS di Departemen Sistem Informasi ITS di masa mendatang.

3.6. Penyusunan Buku Tugas Akhir

Tahap penyusunan buku tugas akhir adalah proses pendokumentasian hasil tugas akhir serta analisis terhadap hasil akhir yang didapatkan. Keluaran dari tahap ini adalah buku tugas akhir. Buku tugas akhir ini diharapkan dapat menjadi referensi bagi penelitian selanjutnya.

BAB IV
PERANCANGAN

Pada bab ini akan dijelaskan bagaimana rancangan dari penelitian tugas akhir yang meliputi subjek dan objek dari tugas akhir, pemilihan subjek dan objek tugas akhir serta bagaimana tugas akhir akan dilakukan.

4.1. Pengumpulan dan Deskripsi Data

Data yang digunakan dalam tugas akhir ini adalah data peserta ujian dan data mata kuliah yang diujikan pada Ujian Tengah Semester (UTS) dan Ujian Akhir Semester (UAS) pada semester genap tahun ajaran 2016 – 2017. Sebagai tambahan, terdapat pula data kurikulum dan data jadwal ujian UTS dan UAS pada semester genap tahun ajaran 2016 – 2017 yang dibuat secara manual. Semua data tersebut diperoleh dari Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

4.1.1. Deskripsi Data Peserta Ujian

Data peserta ujian yang diperoleh merupakan data dalam format Microsoft Excel. Data tersebut mempunyai beberapa kolom, yaitu : periode semester, NRP (nomor identifikasi mahasiswa), nama mahasiswa, kelas mahasiswa, dan kode mata kuliah yang diambil. Gambaran datanya dapat dilihat pada tabel 4.1 .

Tabel 4.1. Sebagian Data Peserta Ujian

No	Semester	NRP	Nama Mahasiswa	Kelas	Kode Mata Kuliah
1	2016.2	5216100703	Athiyatul Ulya	E	KS141205
2	2016.2	5216100703	Athiyatul Ulya	E	KS141207
3	2016.2	5216100703	Athiyatul Ulya	E	KS141208
4	2016.2	5216100162	Maritza Syavira	E	KS141206

No	Semester	NRP	Nama Mahasiswa	Kelas	Kode Mata Kuliah
5	2016.2	5216100162	Maritza Syavira	A	KS141209
6	2016.2	5216100155	Naufal Tsabit	D	KS141207
7	2016.2	5216100155	Naufal Tsabit	A	KS141204
8	2016.2	5216100146	Reza Darmawan	B	KS141205
9	2016.2	5216100146	Reza Darmawan	B	KS141207
10	2016.2	5216100146	Reza Darmawan	B	KS141208

Terdapat dua data peserta ujian, yaitu data peserta Ujian Tengah Semester (UTS) dan data peserta Ujian Akhir Semester (UAS). Kedua data tersebut mempunyai format yang sama seperti yang ditampilkan pada tabel 4.1. Ringkasan dari kedua data tersebut dapat dilihat pada tabel 4.2 .

Tabel 4.2. Ringkasan Data Peserta Ujian

No	Ujian	Jumlah Peserta	Jumlah Mata Kuliah
1	UTS	567	32
2	UAS	519	32

4.1.2. Deskripsi Data Jadwal Ujian

Dalam tugas akhir ini juga diperoleh data jadwal ujian yang dibuat manual. Data ujian manual ini menyimpan informasi mengenai mata kuliah yang diujikan pada semester genap tahun 2016 – 2017, hari dan tanggal ujian, sesi ujian, nama mata kuliah, ruang ujian, sifat ujian (ujian tulis / demo), dan dosen pengampu. Gambaran datanya dapat dilihat pada tabel 4.3 .

Tabel 4.3. Sebagian Jadwal Ujian yang Dibuat Manual

Hari, Tanggal	Sesi	Mata Kuliah	Ruang	Kode Dosen
Senin, 3 April 2017	1	Tata Kelola TI	101, 103, 104, 105, 106, 107	AH, AN
	1	Desain & Manajemen Proses Bisnis	105A	AW
	1	Bahasa Pemrograman	AULA	NF, FA
	2	Pengantar Sistem Operasi	103, 104, 105, 105A, 106	NF, AW
	2	Pemrograman Berbasis Web	101	FS
	3	Manajemen Rantai Pasok dan Hubungan Pelanggan	101, 103, 104, 105, 105A, 106	MW, ES
Selasa 4 April 2017	1	Interaksi Manusia dan Komputer	101, 103, 104, 105, 105A	FJ, RH
	1	Pengantar Basis Data	AULA	RP
	2	Manajemen dan Administrasi Basis Data	101, 102, 103, 104, 105, 106, AULA	RP, FM

Terdapat dua data ujian, yaitu data Ujian Tengah Semester (UTS) dan data Ujian Akhir Semester (UAS). Kedua data tersebut mempunyai format yang sama seperti yang ditampilkan pada tabel 4.3. Ringkasan dari kedua data tersebut dapat dilihat pada tabel 4.4 .

Tabel 4.4. Ringkasan Data Ujian

No	Ujian	Jumlah Mata Kuliah	Jumlah Sesi	Jumlah Ruang
1	UTS	32	15	9
2	UAS	32	15	9

4.1.3. Deskripsi Data Kurikulum

Pada tahun ajaran 2016 – 2017, Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya menggunakan kurikulum tahun 2014 – 2019. Dari data kurikulum tersebut, tugas akhir ini membutuhkan data mata kuliah. Data mata kuliah terdiri dari kode mata kuliah, nama mata kuliah, jumlah SKS mata kuliah, dan periode semester dari tiap mata kuliah. Mata kuliah yang ditawarkan ada dua jenis, yaitu mata kuliah wajib dan mata kuliah pilihan. Secara rinci dapat dilihat pada tabel 4.3 dan 4.4 .

Tabel 4.5. Daftar Mata Kuliah Wajib

No	Semester	Kode	Nama Mata Kuliah	SKS
1	1	IG14110x	Pendidikan Agama	2
2	1	KS141201	Manajemen & Organisasi	3
3	1	KS141106	Wawasan Kebangsaan	3
4	1	KS141202	Sistem & Teknologi Informasi	3
5	1	KS141203	Matematika Diskrit	3
6	1	KS141204	Bahasa Pemrograman	4
7	2	IG141107	Wawasan Teknologi & Komunikasi Ilmiah	3
8	2	KS141205	Algoritma & Struktur Data	3
9	2	IG141108	Bahasa Inggris	3
10	2	KS141206	Pengantar Sistem Operasi	2
11	2	KS141207	Arsitektur SI/TI Perusahaan	3
12	2	KS141208	Kepemimpinan & Ketrampilan Interpersonal	4
13	3	KS141209	Statistika	4
14	3	KS141301	Desain & Manajemen Proses Bisnis	4
15	3	KS141302	Desain & Manajemen Jaringan Komputer	3
16	3	KS141303	Dasar-Dasar Pengembangan Perangkat Lunak	3
17	3	KS141304	Pengantar Basis Data	3

No	Semester	Kode	Nama Mata Kuliah	SKS
18	3	KS141305	Pemrograman Berorientasi Objek	3
19	4	KS141306	Analisis & Desain Perangkat Lunak	4
20	4	KS141307	Interaksi Manusia & Komputer	3
21	4	KS141308	Keamanan Aset Informasi	4
22	4	KS141309	Desain Basis Data	4
23	4	KS141310	Pemrograman Berbasis Web	3
24	5	KS141311	Konstruksi & Pengujian Perangkat Lunak	3
25	5	KS141312	Manajemen Proyek TI	4
26	5	KS141313	Riset Operasi	3
27	5	KS141314	Simulasi Sistem	2
28	5	KS141315	Perencanaan Sumber Daya Perusahaan	4
29	5	KS141316	Manajemen Layanan TI	3
30	6	KS141317	Tata Kelola TI	3
31	6	KS141318	Manajemen Resiko TI	3
32	6	KS141319	Manajemen Pengadaan & Investasi TI	3
33	6	KS141320	Manajemen Rantai Pasok & Hubungan Pelanggan	3
34	6	KS141321	Manajemen & Administrasi Basis Data	3
35	6	KS141322	Sistem Cerdas	3
36	6	KS141210	Tata Tulis Ilmiah	2
37	7	KS141323	Pengukuran Kinerja & Evaluasi TI	3
38	7	KS141324	Kecerdasan Bisnis	3
39	7	KS141325	Etika Profesi SI	2
40	7	IG141109	Technopreneurship	3
41	7	KS141326	Perencanaan Strategis SI/TI	3
42	7	KS1414yz	Kerja Praktik	2
43	7	KS141320	Mata Kuliah Pilihan	3
44	8	KS1414yz	Mata Kuliah Pilihan	6
45	8	KS141320	Tugas Akhir	6

Tabel 4.6. Daftar Mata Kuliah Pilihan

No	Semester	Kode	Nama Mata Kuliah	SKS
1	7 atau 8	KS141401	Sistem Pendukung Keputusan	3
2	7 atau 8	KS141402	Penggalian Data & Analitika Bisnis	3
3	7 atau 8	KS141403	Riset Operasi Lanjut	3
4	7 atau 8	KS141404	Teknik Peramalan	3
5	7 atau 8	KS141405	Pemrogaman Perangkat Bergerak	3
6	7 atau 8	KS141406	Integrasi Aplikasi Korporasi	3
7	7 atau 8	KS141407	Pemrograman Integratif	3
8	7 atau 8	KS141408	Teknologi *Open-Source* dan Terbaru	3
9	7 atau 8	KS141409	*E-Business*	3
10	7 atau 8	KS141410	Komputasi Berpusat pada Manusia	3
11	7 atau 8	KS141411	Forensika Digital	3
12	7 atau 8	KS141412	Perencanaan Keberlangsungan Bisnis	3
13	7 atau 8	KS141413	Manajemen Kualitas SI/TI	3
14	7 atau 8	KS141414	Audit SI	3

4.2. Pra Proses Data

Pada tahap pra proses data, dilakukan pembuatan data yang akan digunakan sebagai *file* masukan untuk aplikasi penjadwalan ujian otomatis. Data tersebut disesuaikan dengan format *Carter dataset.* Format *Carter dataset* terdiri dari dua jenis, yaitu data .crs dan data .stu.

4.2.1. Data CRS

Data dengan format .crs merupakan data yang menyimpan kode mata kuliah dan jumlah peserta mata kuliah. Data ini didapatkan dari data peserta ujian yang telah dijelaskan pada bagian 4.1.1. Pada bagian ini, dilakukan modifikasi terhadap kode mata kuliah. Data kode mata kuliah yang didapatkan dari data

kurikulum, diubah ke dalam format urutan numerik 4 digit. Pemetaan kode mata kuliah dapat dilihat pada tabel 4.7.

Tabel 4.7. Kode Mata Kuliah

No	Kode Mata Kuliah (Awal)	Kode Mata Kuliah (Baru)
1	KS141204	0001
2	KS141205	0002
3	KS141206	0003
4	KS141207	0004
5	KS141208	0005
6	KS141209	0006
7	KS141210	0007
8	KS141301	0008
9	KS141303	0009
10	KS141304	0010
11	KS141306	0011
12	KS141307	0012
13	KS141308	0013
14	KS141309	0014
15	KS141310	0015
16	KS141316	0016
17	KS141317	0017
18	KS141318	0018
19	KS141319	0019
20	KS141320	0020
21	KS141321	0021
22	KS141322	0022
23	KS141323	0023
24	KS141325	0024
25	KS141402	0025
26	KS141403	0026
27	KS141404	0027
28	KS141405	0028
29	KS141407	0029
30	KS141409	0030
31	KS141411	0031
32	KS141413	0032

Data mentah yang sebelumnya disimpan dalam format Microsoft Excel tersebut diolah dengan bantuan *Pivot Table*. Dengan menggunakan *Pivot Table*, dapat diketahui mata kuliah apa saja yang diujikan dan jumlah mahasiswa yang mengambil mata kuliah tersebut. Selanjutnya, data tersebut diubah menjadi format *text file* yang menyimpan dua *fields*, yaitu kode mata kuliah dan jumlah peserta. Kedua *fields* tersebut dipisahkan dengan menggunakan spasi. Lalu *text file* yang sudah terbentuk akan disimpan dalam format .crs. Tampilan data final yang sudah berbentuk format .crs dapat dilihat pada tabel 4.8.

Tabel 4.8. Data .crs

Baris ke-	UTS.crs	UAS.crs
1	0001 4	0001 4
2	0002 155	0002 150
3	0003 150	0003 146
4	0004 154	0004 150
5	0005 152	0005 148
6	0006 64	0006 64
7	0007 157	0007 153
8	0008 17	0008 17
9	0009 28	0009 28
10	0010 1	0010 0
11	0011 160	0011 157
12	0012 150	0012 147
13	0013 161	0013 128
14	0014 159	0014 155
15	0015 144	0015 143
16	0016 21	0016 20
17	0017 190	0017 180
18	0018 176	0018 170
19	0019 138	0019 138
20	0020 172	0020 169
21	0021 184	0021 162
22	0022 191	0022 186
23	0023 28	0023 24
24	0024 30	0024 27
25	0025 27	0025 21
26	0026 28	0026 26
27	0027 40	0027 37

Baris ke-	UTS.crs	UAS.crs
28	0028 4	0028 3
29	0029 18	0029 16
30	0030 50	0030 43
31	0031 38	0031 20
32	0032 40	0032 33

4.2.2. Data STU

Data dengan format .stu merupakan data yang menyimpan mata kuliah yang diambil oleh setiap mahasiswa. Data ini didapatkan dari data peserta ujian yang telah dijelaskan pada bagian 4.1.1. Data mentah yang sebelumnya disimpan dalam format Microsoft Excel tersebut diolah dengan bantuan *Pivot Table*. Dengan menggunakan *Pivot Table*, dapat diketahui mata kuliah apa saja yang diambil oleh setiap mahasiswa dan jumlah mata kuliah yang diambil oleh setiap mahasiswa. Selanjutnya, dilakukan transformasi data menjadi format *text file* yang menyimpan kode mata kuliah yang diambil oleh setiap mahasiswa. Setiap kode mata kuliah tersebut dipisahkan dengan menggunakan spasi. Lalu *text file* yang sudah terbentuk akan disimpan dalam format .stu. Tampilan data final yang sudah berbentuk format .stu dapat dilihat pada tabel 4.9 dan 4.10.

Tabel 4.9. Potongan Data UTS.stu

Baris ke-	UTS.stu
1	0007 0017 0018 0020 0021 0022 0024
2	0007 0017 0018 0020 0021 0022 0026
3	0017 0018 0020 0021 0022 0023
4	0011 0012 0014 0015 0019 0020
5	0017 0018 0019 0021 0022 0028 0029
6	0007 0017 0018 0020 0021 0022 0024 0030
7	0007 0017 0018 0019 0020 0021 0022
8	0012 0014 0017 0018 0019 0020 0022
9	0007 0017 0018 0020 0021 0022 0027
10	0007 0017 0018 0020 0021 0022 0030 0032

Tabel 4.10. Potongan Data UAS.stu

Baris ke-	UAS.stu
1	0007 0017 0018 0019 0020 0021 0022
2	0007 0017 0018 0020 0021 0022 0026 0027
3	0007 0017 0018 0020 0021 0022 0032
4	0017 0018 0019 0020 0021 0022 0025
5	0007 0017 0018 0020 0021 0022 0025 0030
6	0007 0017 0018 0020 0022 0027
7	0017 0018 0019 0020 0021 0022 0030
8	0007 0017 0018 0019 0020 0021 0022
9	0007 0017 0018 0019 0020 0022
10	0011 0013 0014 0018 0020 0022

4.3. Formulasi Fungsi Tujuan

4.3.1. Variabel Keputusan

Variabel keputusan adalah variabel yang harus memenuhi semua *hard constraint* yang telah ditentukan sebelumnya. Variabel keputusan ini digunakan untuk memastikan bahwa setiap mata kuliah harus dijadwalkan. Persamaan matematisnya ditulis pada persamaan (1).

$$X_{it} \begin{cases} 1, & jika\ mata\ kuliah\ ke-i\ dijadwalkan \\ & pada\ timeslot\ ke-t \\ 0, & jika\ tidak \end{cases} \quad (1)$$

Dengan :

X adalah variabel keputusan

$i = \{1,2,3, \dots, n\}$ adalah urutan mata kuliah

$t = \{1,2,3, \dots, k\}$ adalah urutan sesi atau slot ujian

4.3.2. Model Matematis *Hard Constraints*

Hard constraint yang digunakan dalam tugas akhir ini ada tiga. Pertama, semua mata kuliah harus dijadwalkan. Kedua, tidak boleh ada jadwal ujian yang bentrok antar mata kuliah. Ketiga, total mahasiswa yang mengikuti ujian tidak melebihi kapasitas total ruang ujian yang tersedia.

Untuk memastikan bahwa jadwal yang dibuat telah memenuhi *hard constraint* dimana setiap mata kuliah harus dijadwalkan, maka dinyatakan dengan persamaan matematis (2).

$$\sum_{i=1}^{n}\sum_{t=1}^{k} X_{it} = 1 \quad (2)$$

Dimana X_{it} adalah mata kuliah ke-i yang dijadwalkan pada slot ujian ke-t. Sedangkan untuk memastikan bahwa jadwal yang dibuat telah memenuhi *hard constraint* kedua dimana tidak ada mata kuliah yang mengalami bentrok dinyatakan dengan persamaan matematis (3).

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} C_{ij}.V_{ij} = 0$$

$$V_{ij}\begin{cases} 1\ jika\ t_i = t_j \\ 0 \end{cases} \quad (3)$$

Dengan :
C, V adalah variabel keputusan
$i, j = \{1,2,3,\dots,n\}$ adalah urutan mata kuliah
C_{ij}, V_{ij} = jumlah mahasiswa peserta mata kuliah i dan j
t_i = slot ujian dimana mata kuliah ke i dijadwalkan

Untuk memastikan bahwa jadwal yang dibuat telah memenuhi *hard constraint* ketiga dimana total mahasiswa peserta ujian tidak melebihi kapasitas total ruang ujian pada slot ujian ke-j maka dinyatakan dengan persamaan matematis (4).

$$\sum_{i=1}^{n} S_i X_{it} \leq kapasitas \quad (4)$$

Dengan :
$i = \{1,2,3,\dots,n\}$ adalah urutan mata kuliah
S_i = jumlah peserta ujian untuk mata kuliah ke-i

4.3.3. Fungsi Tujuan

Fungsi tujuan yang digunakan untuk memastikan bahwa jadwal yang dihasilkan telah optimal atau tidak adalah dengan cara

meminimalkan nilai *proximity cost* (P) yang dinyatakan dalam persamaan matematis (5).

$$P = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} C_{ij} W_{|t_j - t_i|}}{M}$$

$$W_{|t_i - t_j|} = \begin{cases} 2^{5-|t_i - t_j|} \; jika \; 1 \leq |t_i - t_j| \leq 5 \\ 0 \; jika \; |t_i - t_j| > 5 \end{cases} \quad (5)$$

Dengan :

$i, j = \{1,2,3, \ldots, n\}$ adalah urutan mata kuliah

t_i = slot ujian untuk mata kuliah i

C_{ij} = jumlah peserta ujian mata kuliah i dan mata kuliah j

$W_{|t_j - t_i|}$ = bobot selisih slot ujian mata kuliah i dan j

M = jumlah seluruh mahasiswa.

4.3.4. Pembentukan *Conflict Matrix*

Setelah *input file* .crs dan .stu terbentuk, maka langkah selanjutnya adalah membuat *conflict matrix. Conflict Matrix* adalah matriks yang menyimpan nilai banyaknya mahasiswa yang mengambil dua mata kuliah secara bersamaan. Baris dan kolom pada matriks menunjukkan kode mata kuliah. *Conflict Matrix* secara keseluruhan dapat dilihat pada Lampiran A. Tabel 4.12 menunjukkan potongan dari *conflict matrix* tersebut. Untuk mengetahui nilai C_{ij}, data mahasiswa yang telah didapatkan perlu diubah kedalam sebuah matriks dua dimensi dengan jumlah kolom dan baris sebanyak 32 sesuai dengan jumlah mata kuliah, sehingga didapatkan matriks seperti pada tabel 4.12 dimana setiap sel dari tabel tersebut berisi jumlah mahasiswa yang mengikuti mata kuliah C_{ij}. *S*elain itu fungsi matriks ini juga untuk mengetahui mata kuliah mana saja yang berpotensi bertabrakan pada saat dijadwalkan.

Tabel 4.11 Sebagian *Conflict Matrix*

	0001	**0002**	**0003**	**0004**	**0005**	**0006**	**0007**	**0008**
0001	0	4	4	4	4	0	0	0
0002	4	0	150	152	151	59	1	8
0003	4	150	0	150	150	57	0	7

0004	4	152	150	0	151	58	1	8
0005	4	151	150	151	0	57	0	7
0006	0	59	57	58	57	0	1	3
0007	0	1	0	1	0	1	0	1
0008	0	8	7	8	7	3	1	0

4.4. Penerapan Algoritma *Greedy*

Algoritma *greedy* digunakan untuk menghasilkan jadwal baru berdasarkan data masukan yang telah dipraproses terlebih dahulu. Jadwal yang dihasilkan pada tahapan ini akan menjadi masukan yang akan dioptimasi oleh algoritma *late acceptance.*

4.5. Penerapan Algoritma *Late acceptance*

Penerapan algoritma *late acceptance* ditujukan untuk mengoptimasi jadwal yang telah dihasilkan oleh algoritma *greedy* pada tahap sebelumnya. Pada tahap ini optimasi dilakukan dengan menukar atau mengganti slot dari salah satu mata kuliah.

Halaman ini sengaja dikosongkan.

BAB V
IMPLEMENTASI

Pada bab ini berisi tentang proses implementasi algoritma *greedy – late acceptance – hyper heuristic* dalam mencari hasil penjadwalan ujian yang optimum dari studi kasus tugas akhir dengan menggunakan bahasa pemrograman *Java* dan *tools* NetBeans IDE 8.2. Implementasi yang dilakukan terdiri dari beberapa *method* sebagai berikut.

5.1. Membaca *Input File*

Langkah pertama yang dilakukan dalam implementasi aplikasi penjadwalan ujian otomatis adalah membaca *input file.* Terdapat dua jenis *input file*, yaitu *file* .crs dan .stu yang telah dijelaskan pada bagian 4.2.

```
void bacaCRSSTU() {
    try {

        FileReader fileReaderCRS = new FileReader(FileCRS);
        LineNumberReader lineReaderCRS = new LineNumberReader(fileReaderCRS);
        String lineTextCRS = null;

        while ((lineTextCRS = lineReaderCRS.readLine()) != null) {

            String[] aryCRS = lineTextCRS.split(" ");

            String JumlahMhs = aryCRS[1];
            int JumlahMhsInt = Integer.parseInt(JumlahMhs);
            JumlahMhsPerMK.add(JumlahMhsInt);

            TotalMK++;
        }
        setTotalMK(TotalMK);

        lineReaderCRS.close();
    } catch (Exception e) {
```

Kode 5.1 *Method* Baca File .crs

Implementasi pembacaan file .crs dapat dilihat pada potongan kode 5.1 dimana pembacaan ini bertujuan supaya sistem penjadwalan dapat mengetahui jumlah mata kuliah dan jumlah mahasiswa pada setiap mata kuliah. Untuk file .stu, pembacaan dilakukan supaya sistem mengetahui mata kuliah apa saja yang

diambil oleh setiap mahasiswa. Kemudian setiap mata kuliah tersebut akan disimpan didalam *array.*

```
try {
    FileReader fileReaderSTU = new FileReader(FileSTU);
    LineNumberReader lineReaderSTU = new LineNumberReader(fileReaderSTU);
    String lineTextSTU = null;
    int barisUrut = 0;
    while ((lineTextSTU = lineReaderSTU.readLine()) != null) {
        String[] arySTU = lineTextSTU.split(" ");
        STU[barisUrut] = new int[arySTU.length];
        for (int i = 0; i < arySTU.length; i++) {
            STU[barisUrut][i] = Integer.parseInt(arySTU[i]);
        }

        barisUrut++;

    }
    lineReaderSTU.close();
} catch (Exception e) {
```

Kode 5.2 *Method* Baca File .stu

5.2. Pembuatan *Conflict matrix*

Pembuatan *conflict matrix* dilakukan untuk mengetahui jumlah mahasiswa yang mengikuti dua mata kuliah. *Conflict matrix* terdiri dari *array* dua dimensi yang menjadi kolom dan baris. Panjang kolom dan baris dari *conflict matrix* ini bergantung dari jumlah mata kuliah hasil pembacaan dari file .stu.

```
int[][] CM() {
    int[][] conflictMatrix = new int[TotalMK][TotalMK];
    for (int i = 0; i < getSTU().length; i++) {
        for (int j = 0; j < getSTU()[i].length - 1; j++) {
            for (int k = j + 1; k < getSTU()[i].length; k++) {
                int arySTi = getSTU()[i][j];
                int arySTj = getSTU()[i][k];
                conflictMatrix[arySTi - 1][arySTj - 1]++;
                conflictMatrix[arySTj - 1][arySTi - 1]++;
            }
        }
    }
    return conflictMatrix;
```

Kode 5.3 *Method Conflict Matrix*

5.3. Penerapan Algoritma *Greedy*

Algoritma *greedy* diterapkan pada pengerjaan tugas akhir ini untuk menghasilkan jadwal ujian awal yang nantinya menjadi input untuk optimasi yang dilakukan oleh algoritma *late acceptance*. Untuk menentukan suatu ujian dapat dilaksanakan pada suatu slot waktu digunakan dua macam *method* yaitu okToSlot dan okToRoom. Pada okToSlot mata kuliah kedua akan dibandingkan dengan mata kuliah sebelumnya untuk melihat apakah terdapat bentrokan.

```
boolean okToSlot(int lecture, int slot, int[][] schd) {
    for (int i = 0; i < schd[lecture].length; i++) {
        int ithAdjLecture = schd[lecture][i];
        if (subjectTimeSlot[ithAdjLecture] == slot) {
            return false;
        }
    }
    return true;
}
```

Kode 5.4 *Method* okToSlot

Berikutnya *method* okToRoom digunakan untuk membandingkan jumlah mahasiswa peserta suatu mata kuliah dengan jumlah kapasitas ruangan yang tersisa yang apabila peserta suatu mata kuliah lebih kecil jumlahnya dari kapasitas pada suatu slot maka akan mata kuliah tersebut akan dimasukkan kedalam slot tersebut dan sisa kapasitas pada slot tersebut akan dikurangi dengan jumlah mahasiswa peserta mata kuliah tersebut.

```
boolean okToRoom(int index, int session, int[] AmountStudentPerSubject) {
    int a = AmountStudentPerSubject[index];
    if (a <= remaincapacity[session - 1]) {
        remaincapacity[session - 1] = remaincapacity[session - 1] - a;
        return true;
    }

    return false;
}
```

Kode 5.5 *Method* okToRoom

Apabila suatu mata kuliah telah memenuhi kedua *method* diatas maka mata kuliah tersebut dapat dimasukkan kedalam suatu slot dengan *method* explore, namun apabila suatu mata kuliah tidak memenuhi salah satu atau kedua *method* maka akan dipindahkan ke slot berkutnya dan akan.kembali dicocokkan apakah memenuhi kondisi pada *method* okToRoom dan okToSlot.

```
boolean explore(int lecture, int slot, int[][] schd, int[] AmountStudentPerSubject) {
    if (lecture >= schd.length) {
        return true;
    }
    int realMK = MK.get(lecture).getIndeks();
    if (okToSlot(realMK, slot, schd) && okToRoom(realMK, slot, AmountStudentPerSubject)) {
        subjectTimeSlot[realMK] = slot;
        for (int i = 1; i <= remaincapacity.length; i++) {
            if (explore(lecture + 1, i, schd, AmountStudentPerSubject)) {
                return true;
            }
        }
    }

    return false;
}
```

Kode 5.6 *Method* Untuk Menetukan Slot Ujian

5.4. Penerapan Algoritma *Late acceptance*

Penerapan algoritma *late acceptance* bertujuan untuk mengoptimasi jadwal yang telah didapat dari algoritma *greedy*. Algoritma *late acceptance* juga diharapkan dapat menghasilkan nilai *proximity cost* yang lebih rendah dibandingkan dengan nilai *proximity cost* yang didapatkan pada algoritma *greedy*. Pada tugas akhir ini, penerapan algoritma *late acceptance* dibagi menjadi dua, yaitu berdasarkan jumlah iterasi dan berdasarkan waktu program dijalankan. Penjelasan lebih lengkapnya dapat dilihat pada bagian 5.4.1 dan 5.4.2.

5.4.1. Algoritma *Late acceptance* Berdasarkan Jumlah Iterasi

Sebelum dapat menjalankan algoritma ini, beberapa variabel perlu diinisiasi seperti sebuah *LinkedList* yang dilambangkan dengan *antrianLate*, nilai *proximity cost* terbaik yang dilambangkan dengan *bestSol*, dan sebuah *array* untuk

menyimpan solusi terbaik yang dilambangkan dengan *bestSolArray*. Kode 5.7 menunjukkan inisiasi variabel untuk *method late acceptance*.

```
void LAHC(int late, int jmlhLoop) {
        LinkedList<Double> antrianLate = new LinkedList<Double>();
        double bestSol = hitungProximity(subjectTimeSlot);
        int[] bestSolArray = subjectTimeSlot.clone();

        for (int i = 0; i < late; i++) {
            antrianLate.addLast(hitungProximity(subjectTimeSlot));
        }
```

Kode 5.7 Inisiasi Variabel untuk Algoritma *Late acceptance*

Jadwal ujian yang telah dihasilkan dari algoritma *greedy* akan dimodifikasi untuk memperoleh jadwal yang lebih optimal. Untuk memodifikasi jadwal tersebut dilakukan dua cara, yaitu mengubah *(move)* salah satu slot ujian menjadi slot yang berbeda atau menukar (*swap*) dua slot ujian. Pada *method move*, akan dipilih salah satu slot ujian secara acak. Lalu slot tersebut akan diganti menjadi slot lain secara acak pula. Setelah itu, jadwal akan kembali dibandingkan dengan menggunakan *method* okToRoom dan okToSlot untuk memastikan bahwa jadwal yang baru telah memenuhi dua *method* tersebut. Pada *method swap*, akan dipilih dua slot ujian secara acak. Lalu kedua slot ujian tersebut akan ditukar dan kemudian akan kembali dibandingkan dengan menggunakan *method* okToRoom dan okToSlot untuk memastikan bahwa jadwal yang baru telah memenuhi dua *method* tersebut. Modifikasi jadwal menggunakan *move* dan *swap* tersebut akan dilakukan secara acak. Kode 5.8 menunjukkan metode acak yang digunakan saat memilih *method move* atau *swap*.

```
        for (int i=1; i <= jmlhLoop; i++){
            System.out.println("Iterasi ke-" + i + ":");
            if (Math.random() < 0.5) {
                move();
            } else {
                swap();
            }
```

Kode 5.8. Random Move & Swap untuk Late Acceptance

Selanjutnya, mulai menjalankan algoritma *late acceptance*. *Method late acceptance* menggunakan dua parameter, yaitu jumlah antrian dan jumlah iterasi. Algoritma *late acceptance* memerlukan variabel berupa *LinkedList* dengan sistem *FIFO (first in first out)* untuk menyimpan sejumlah antrian. *Method late acceptance* ini akan membuat panjang antrian sesuai dengan parameter yang telah diatur serta menjalankan iterasi sejumlah parameter yang telah diatur pula. Pada iterasi pertama, jadwal ujian yang dihasilkan dari algoritma *greedy* akan dimasukkan ke dalam antrian. Selanjutnya, jadwal akan dimodifikasi dengan method *move* atau *swap*. Jadwal baru yang terbentuk tersebut akan disimpan ke dalam antrian, namun hanya sejumlah parameter yang telah diatur. Kode 5.9 menunjukkan *method late acceptance* yang digunakan.

```
            double newOF = hitungProximity(jadwalBaru);
            antrianLate.removeFirst();
            antrianLate.addLast(newOF);
            System.out.println(newOF + " " + bestSol);
            int id = i % late;
            if (newOF < bestSol) {
                bestSolArray = jadwalBaru.clone();
                bestSol = newOF;
            }
            double comparator = antrianLate.get(id);
            if (newOF < comparator) {
                jadwalAwal = jadwalBaru;
            }
            System.out.println("");
```

Kode 5.9. *Method Late Acceptance*

Jika nilai *proximity cost* dari jadwal yang berada di antrian tersebut lebih baik dari nilai *proximity cost* untuk jadwal yang diperoleh dari algoritma *greedy*, maka jadwal tersebut akan menggantikan jadwal yang semula. Selanjutnya, nilai *proximity cost* dari jadwal baru tersebut akan digunakan sebagai pembanding di iterasi berikutnya.

Untuk mengetahui nilai *proximity cost* terendah yang didapatkan pada setiap iterasi, maka ditampilkan nilai *bestSol*. Kode 5.10 menunjukkan kode untuk menampilkan nilai *bestSol*

yang merupakan nilai *proximity cost* terendah yang berhasil didapatkan.

```
        System.out.println("best sol = " + bestSol);
        for (int j = 0; j < bestSolArray.length; j++) {
            System.out.print(bestSolArray[j] + " ");
        }
        System.out.println("\n");
```

Kode 5.10. Cetak *Proximity Cost*

5.4.2. Algoritma *Late acceptance* Berdasarkan Waktu

Selain menggunakan parameter jumlah iterasi, penerapan algoritma *late acceptance* juga dapat dijalankan berdasarkan satuan waktu (menit). Secara keseluruhan, kode program yang digunakan hampir sama dengan bagian 5.4.1. Jumlah iterasi yang digunakan pada bagian 5.4.1 diubah menjadi satuan waktu. Sehingga parameter yang diatur disini adalah jumlah antrian dan batas waktu program dijalankan dalam satuan menit. Kode 5.11 menunjukkan *method late acceptance* yang dijalankan berdasarkan satuan waktu.

```
    void LATime(int late, int time) {
        final long NANOSEC_PER_SEC = 10001 * 1000 * 1000;
        long startTime = System.nanoTime();
        LinkedList<Double> antrianLate = new LinkedList<Double>();
        double bestSol = hitungProximity(subjectTimeSlot);
        int[] bestSolArray = subjectTimeSlot.clone();

        for (int i = 0; i < late; i++) {
            antrianLate.addLast(hitungProximity(subjectTimeSlot));
        }

        int B=0;

        while ((System.nanoTime() - startTime) < time * 60 * NANOSEC_PER_SEC) {
            if (Math.random() < 0.5) {
                move();
            } else {
                swap();
            }
```

Kode 5.11. *Method Late acceptance* Berdasarkan Waktu

Halaman ini sengaja dikosongkan.

BAB VI
HASIL DAN PEMBAHASAN

Bab ini akan menjelaskan hasil yang didapatkan dari penelitian ini dan pembahasan secara keseluruhan yang didapatkan dari penelitian.

6.1. Data Uji Coba

Data yang digunakan adalah data peserta ujian serta mata kuliah tahun ajaran 2016-2017 Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember. Pada data ini diketahui terdapat 567 mahasiswa dan 32 mata kuliah.

6.2. Lingkungan Uji Coba

Lingkungan uji coba merupakan kriteria perangkat pengujian yang digunakan untuk implementasi penjadwalan ujian otomatis dengan menggunakan algoritma *greedy – late acceptance – hyper heuristic* dalam penyelesaian permasalahan pada tugas akhir ini. Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang digunakan. Spesifikasi dari perangkat keras yang digunakan dalam implementasi metode ditunjukkan pada Tabel 6.1.

Tabel 6.1 Lingkungan Uji Coba Perangkat Keras

Perangkat Keras	Spesifikasi
Jenis	Notebook
Processor	Intel® Core™ i7 4510U CPU @ 2.00 GHz 2.60 GHz
RAM	4.00 GB (3.89 usable)
System type	64-bit Operating System, x64-based processor
Hard Disk Drive	1 TB

Selain perangkat keras, terdapat lingkungan perangkat lunak yang digunakan dalam implementasi metode seperti yang ditunjukkan pada Tabel 6.2.

Tabel 6.2 Lingkungan Uji Coba Perangkat Lunak

Perangkat Lunak	Spesifikasi
Sistem Operasi	Windows 10
Bahasa Pemrograman	Java
Tools	Netbeans IDE 8.2 Microsoft Excel 2016

6.3. Fungsi Tujuan dari Jadwal Manual

Fungsi tujuan yang digunakan dalam tugas akhir ini adalah *proximity cost.* Sebelum membuat jadwal ujian otomatis, dilakukan perhitungan *proximity cost* terhadap jadwal ujian yang telah ada (jadwal ujian yang dibuat manual) dengan menggunakan *microsoft excel* berdasarkan persamaan berikut.

$$P = \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} C_{ij} W_{|t_j - t_i|}}{S}$$

Dari hasil perhitungan, diketahui bahwa nilai *proximity cost* untuk jadwal ujian yang dibuat secara manual sebesar 39,569 untuk UTS dan 76,763 untuk UAS. Namun, jadwal ujian yang dibuat manual tersebut masih terdapat beberapa ujian yang bentrok.

Setelah dilakukan perhitungan ulang dengan menggunakan sistem yang dibangun dengan bahasa pemrograman *java* didapati pada jadwal UTS manual terdapat dua mata kuliah yang mengalami bentrok jadwal dan kelebihan kapasitas yang ditandai dengan peringatan *NOT FEASIBLE* pada luaran yang dihasilkan. Untuk bentrok jadwal terjadi pada mata kuliah dengan kode KS141210 dan KS14108. Jadwal yang bentrok ini menyebabkan 3 orang mahasiswa tidak dapat mengikuti ujian dengan baik. Sedangkan untuk kasus kelebihan kapasitas pada hari Senin slot ke 2 dan hari Jumat slot 2 apabila seluruh ujian

dilaksanakan di ruang kelas. Gambar 6.1 dan 6.2 menunjukkan tampilan hasil program yang d

```
NOT FEASIBLE 6 12 Sebanyak 3 mahasiswa
NOT FEASIBLE ROOM CAPACITY EXCEEDED: Timeslot ke 2
NOT FEASIBLE ROOM CAPACITY EXCEEDED: Timeslot ke 14
```

Gambar 6.1 Permasalahan pada Jadwal UTS Manual

```
NOT FEASIBLE 1 10 Sebanyak 1 mahasiswa
NOT FEASIBLE 6 12 Sebanyak 2 mahasiswa
NOT FEASIBLE 6 21 Sebanyak 144 mahasiswa
NOT FEASIBLE 8 12 Sebanyak 1 mahasiswa
NOT FEASIBLE 8 21 Sebanyak 1 mahasiswa
NOT FEASIBLE 10 22 Sebanyak 1 mahasiswa
NOT FEASIBLE 11 13 Sebanyak 141 mahasiswa
NOT FEASIBLE 11 17 Sebanyak 3 mahasiswa
NOT FEASIBLE 12 21 Sebanyak 9 mahasiswa
NOT FEASIBLE 13 17 Sebanyak 7 mahasiswa
NOT FEASIBLE 17 24 Sebanyak 17 mahasiswa
NOT FEASIBLE 22 25 Sebanyak 2 mahasiswa
NOT FEASIBLE 29 30 Sebanyak 8 mahasiswa
NOT FEASIBLE 29 31 Sebanyak 3 mahasiswa
NOT FEASIBLE 30 31 Sebanyak 5 mahasiswa
```

Gambar 6.2 Permasalahan pada Jadwal UAS Manual

Pada jadwal UAS manual diketahui bahwa nilai *proximity cost* yaitu sebesar 76,763. Setelah dilakukan perhitungan ulang nilai dengan mnggunakan sistem yang dibangun dengan bahasa *java* didapati nilai *proximity cost* sebesar ini disebabkan oleh banyaknya jadwal ujian yang saling bertabrakan. Banyaknya jumlah ujian yang saling bertabrakan ini disebabkan jumlah slot ujian yang digunakan hanya terdapat 12 slot ujian dari 15 slot waktu ujian dikarenakan pada minggu ujian terpotong satu hari untuk hari libur nasional. Pada tabel 6.3 dapat dilihat daftar mata kuliah yang jadwal ujiannya bentrok.

Tabel 6.3 Mata Kuliah dengan Jadwal Bentrok

Mata Kuliah 1	Mata Kuliah 2	Jumlah Mahasiswa
KS141205	KS141306	1
KS141306	KS141308	2
KS141306	KS141322	144
KS141303	KS141308	1
KS141303	KS141322	1
KS141306	KS141323	1
KS141307	KS141309	141
KS141307	KS141318	3
KS141308	KS141322	9
KS141309	KS141318	7
KS141318	KS141402	17
KS141323	KS141403	2

Dapat disimpulkan bahwa jadwal ujian yang dibuat manual tersebut dianggap gagal memenuhi *hard constraint* yang ditetapkan, yaitu tidak boleh ada ujian yang bentrok dan tidak boleh melebihi kapasitas total ruangan. Sehingga nilai *proximity cost* awal yang didapat harus mendapat penalti karena melanggar *hard constraints*. Nilai *proximity cost* baru untuk jadwal ujian manual adalah 208.712 untuk UTS dan 19.521.693 untuk UAS. Nilai *proximity cost* ini akan menjadi pembanding terhadap hasil *proximity cost* yang dihasilkan dengan menggunakan sistem penjadwal otomatis.

6.4. Hasil Penerapan Algoritma *Greedy*

Algoritma *greedy* digunakan untuk menghasilkan solusi awal yang *feasible.* Solusi awal tersebut adalah solusi yang telah memenuhi batasan dimana tidak ada jadwal ujian yang bentrok dan tidak melebihi total kapasitas ruang. Tabel 6.4 dan 6.5 menunjukkan hasil jadwal ujian yang diperoleh dari penerapan algoritma *greedy.*

Tabel 6.4 Jadwal *Greedy*

Kode Mata Kuliah	UTS	UAS
0001	1	2
0002	2	11
0003	14	13
0004	11	1
0005	13	12
0006	5	4
0007	3	2
0008	6	5
0009	8	6
0010	1	1
0011	4	8
0012	3	2
0013	1	3
0014	7	10
0015	12	9
0016	11	11
0017	6	4
0018	5	5
0019	2	1
0020	8	6
0021	10	9
0022	9	7
0023	7	10
0024	11	11
0025	12	3
0026	13	3
0027	14	10
0028	7	3
0029	1	12
0030	4	8
0031	15	13
0032	12	12

Jadwal yang dihasilkan oleh algoritma *greedy* memiliki nilai *proximity cost* sebesar 60,885 untuk UTS dan 75,079 untuk UAS. Hal ini menunjukkan bahwa hasil jadwal ujian yang menjadi luaran algoritma *greedy* masih mempunyai nilai

proximity cost yang cukup besar. Tetapi, jadwal yang dihasilkan oleh algoritma *greedy* ini sudah *feasible* sehingga dapat dikatakan bahwa jadwal ujian yang dihasilkan oleh algoritma *greedy* adalah jadwal ujian yang lebih baik daripada jadwal ujian yang dibuat secara manual. Selanjutnya, jadwal ujian yang dihasilkan oleh algoritma *greedy* ini masih perlu untuk dioptimalisasi dengan menggunakan algoritma *late acceptance* sehingga nilai *proximity cost* yang diperoleh akan semakin baik.

6.5. Hasil Penerapan Algoritma *Late acceptance*

Algoritma *late acceptance* digunakan untuk mengoptimasi jadwal ujian hasil dari penerapan algoritma *greedy*. Untuk mencari nilai *proximity cost* terbaik, maka dilakukan beberapa uji coba dengan mengubah parameter yang digunakan dalam algoritma *late acceptance*.

6.5.1. Skenario Uji Coba 1

Skenario uji coba yang pertama mengambil referensi dari penelitian yang telah dilakukan sebelumnya. Menurut penelitian yang dilakukan oleh Anas Arram, Masri Ayob, dan Mohammad Zakree yang menggunakan algoritma *late acceptance* untuk memecahkan *traveling salesman problem (TSP)* [16], parameter yang digunakan dalam penelitian tersebut adalah jumlah antrian sebanyak 5.000 dan jumlah iterasi sebanyak 1.000.000 kali. Oleh karena itulah, pada uji coba pertama ini, parameter yang digunakan akan disesuaikan dengan referensi tersebut.

Untuk mencari nilai *proximity cost* yang terbaik, dilakukan 11 kali uji coba. Dalam 11 kali percobaan, nilai *proximity cost* minimum yang berhasil didapatkan yaitu sebesar 41,252 dan nilai *proximity cost* maksimum yang berhasil didapatkan yaitu sebesar 52,457. Sedangkan rata-ratanya sebesar 46,806.

Selanjutnya, nilai *proximity cost* yang didapat dari penerapan algoritma *late acceptance* tersebut dapat dibandingkan dengan nilai *proximity cost* dari jadwal yang dibuat secara manual dan jadwal yang diperoleh dari penerapan algoritma *greedy*. Perlu

diingat bahwa jadwal manual adalah jadwal yang tidak *feasible* sehingga nilai *proximity cost*nya pun sangat besar, yaitu senilai 208.712. Di samping itu, jadwal *greedy* adalah jadwal yang sudah *feasible* tetapi belum optimal. Sehingga nilai *proximity cost* untuk jadwal *greedy* juga masih cukup besar, yaitu senilai 60,885. Sedangkan untuk jadwal *late acceptance*, jadwal ini adalah jadwal yang tidak hanya *feasible* tetapi juga sudah dioptimalkan. Sehingga nilai *proximity cost* yang didapatkan dari jadwal ini bisa turun ke angka 41,252. Perubahan nilai *proximity cost* dari jadwal *greedy* ke jadwal *late acceptance* dinilai cukup signifikan, yaitu sebesar 19,633. Tabel 6.5 menunjukkan jadwal hasil uji coba pertama dengan nilai *proximity cost* terkecil.

Tabel 6.5. Jadwal Hasil Uji Coba 1

Kode Mata Kuliah	Sesi ke-
0001	6
0002	11
0003	2
0004	5
0005	14
0006	7
0007	3
0008	8
0009	10
0010	14
0011	13
0012	3
0013	6
0014	9
0015	12
0016	14
0017	7

Kode Mata Kuliah	Sesi ke-
0018	15
0019	2
0020	1
0021	10
0022	8
0023	9
0024	5
0025	12
0026	14
0027	11
0028	12
0029	6
0030	4
0031	13
0032	12

6.5.2. Skenario Uji Coba 2

Skenario uji coba yang kedua masih mengambil referensi dari penelitian yang telah dilakukan sebelumnya. Menurut penelitian yang dilakukan oleh Anas Arram, Masri Ayob, dan Mohammad Zakree yang menggunakan algoritma *late acceptance* untuk memecahkan *traveling salesman problem (TSP)* [16], parameter yang digunakan dalam penelitian tersebut adalah jumlah antrian sebanyak 5.000 dan jumlah iterasi sebanyak 1.000.000 kali. Namun, kali ini *method* yang digunakan adalah hanya *method move* saja, tanpa *swap*.

Untuk mencari nilai *proximity cost* yang terbaik, dilakukan 11 kali uji coba. Dalam 11 kali percobaan, nilai *proximity cost* minimum yang berhasil didapatkan yaitu sebesar 50,792 dan nilai *proximity cost* maksimum yang berhasil didapatkan yaitu sebesar 59,111. Sedangkan rata-ratanya sebesar 56,157.

Selanjutnya, nilai *proximity cost* yang didapat dari penerapan algoritma *late acceptance* tersebut dapat dibandingkan dengan nilai *proximity cost* dari jadwal yang dibuat secara manual dan jadwal yang diperoleh dari penerapan algoritma *greedy*. Perlu diingat bahwa jadwal manual adalah jadwal yang tidak *feasible* sehingga nilai *proximity cost*nya pun sangat besar, yaitu senilai 208.712. Di samping itu, jadwal *greedy* adalah jadwal yang sudah *feasible* tetapi belum optimal. Sehingga nilai *proximity cost* untuk jadwal *greedy* juga masih cukup besar, yaitu senilai 60,885. Sedangkan untuk jadwal *late acceptance*, jadwal ini adalah jadwal yang tidak hanya *feasible* tetapi juga sudah dioptimalkan. Sehingga nilai *proximity cost* yang didapatkan dari jadwal ini bisa turun ke angka 50,792. Perubahan nilai *proximity cost* dari jadwal *greedy* ke jadwal *late acceptance* dinilai cukup signifikan, yaitu sebesar 10,093. Tabel 6.6 menunjukkan jadwal hasil uji coba kedua dengan nilai *proximity cost* terkecil.

Tabel 6.6. Hasil Uji Coba 2

Kode Mata Kuliah	Sesi ke-
0001	4
0002	2
0003	7
0004	11
0005	14
0006	5
0007	3
0008	9
0009	3
0010	14
0011	4
0012	13
0013	1
0014	7

Kode Mata Kuliah	Sesi ke-
0015	12
0016	11
0017	6
0018	5
0019	2
0020	8
0021	10
0022	9
0023	14
0024	11
0025	12
0026	13
0027	14
0028	14
0029	1
0030	4
0031	15
0032	12

6.5.3. Skenario Uji Coba 3

Skenario uji coba yang ketiga masih mengambil referensi dari penelitian yang telah dilakukan sebelumnya. Menurut penelitian yang dilakukan oleh Anas Arram, Masri Ayob, dan Mohammad Zakree yang menggunakan algoritma *late acceptance* untuk memecahkan *traveling salesman problem (TSP)* [16], parameter yang digunakan dalam penelitian tersebut adalah jumlah antrian sebanyak 5.000 dan jumlah iterasi sebanyak 1.000.000 kali. Namun, kali ini *method* yang digunakan adalah hanya *method swap* saja, tanpa *move.*

Untuk mencari nilai *proximity cost* yang terbaik, dilakukan 11 kali uji coba. Dalam 11 kali percobaan, nilai *proximity cost*

minimum yang berhasil didapatkan yaitu sebesar 40,801 dan nilai *proximity cost* maksimum yang berhasil didapatkan yaitu sebesar 49,910. Sedangkan rata-ratanya sebesar 46,004.

Selanjutnya, nilai *proximity cost* yang didapat dari penerapan algoritma *late acceptance* tersebut dapat dibandingkan dengan nilai *proximity cost* dari jadwal yang dibuat secara manual dan jadwal yang diperoleh dari penerapan algoritma *greedy*. Perlu diingat bahwa jadwal manual adalah jadwal yang tidak *feasible* sehingga nilai *proximity cost*nya pun sangat besar, yaitu senilai 208.712. Di samping itu, jadwal *greedy* adalah jadwal yang sudah *feasible* tetapi belum optimal. Sehingga nilai *proximity cost* untuk jadwal *greedy* juga masih cukup besar, yaitu senilai 60,885. Sedangkan untuk jadwal *late acceptance*, jadwal ini adalah jadwal yang tidak hanya *feasible* tetapi juga sudah dioptimalkan. Sehingga nilai *proximity cost* yang didapatkan dari jadwal ini bisa turun ke angka 40,801. Perubahan nilai *proximity cost* dari jadwal *greedy* ke jadwal *late acceptance* dinilai cukup signifikan, yaitu sebesar 20,084. Tabel 6.7 menunjukkan jadwal hasil uji coba ketiga dengan nilai *proximity cost* terkecil.

Tabel 6.7. Hasil Uji Coba 3

Kode Mata Kuliah	Sesi ke-
0001	1
0002	2
0003	6
0004	11
0005	8
0006	14
0007	3
0008	5
0009	13
0010	3
0011	4

Kode Mata Kuliah	Sesi ke-
0012	1
0013	15
0014	7
0015	12
0016	11
0017	9
0018	5
0019	2
0020	14
0021	13
0022	10
0023	7
0024	8
0025	12
0026	11
0027	1
0028	7
0029	1
0030	6
0031	4
0032	12

6.5.4. Skenario Uji Coba 4

Skenario uji coba yang keempat dilakukan dengan menjalankan *method move* dan *swap* secara acak. Dalam uji coba keempat, jumlah antrian diubah menjadi 100. Sedangkan jumlah iterasi dilakukan sebanyak 1.000.000 kali.

Untuk mencari nilai *proximity cost* yang terbaik, dilakukan 11 kali uji coba. Dalam 11 kali percobaan, nilai *proximity cost* minimum yang berhasil didapatkan yaitu sebesar 38,882 dan

nilai *proximity cost* maksimum yang berhasil didapatkan yaitu sebesar 47,660. Sedangkan rata-ratanya sebesar 44,654.
Selanjutnya, nilai *proximity cost* yang didapat dari penerapan algoritma *late acceptance* tersebut dapat dibandingkan dengan nilai *proximity cost* dari jadwal yang dibuat secara manual dan jadwal yang diperoleh dari penerapan algoritma *greedy*. Perlu diingat bahwa jadwal manual adalah jadwal yang tidak *feasible* sehingga nilai *proximity cost*nya pun sangat besar, yaitu senilai 208.712. Di samping itu, jadwal *greedy* adalah jadwal yang sudah *feasible* tetapi belum optimal. Sehingga nilai *proximity cost* untuk jadwal *greedy* juga masih cukup besar, yaitu senilai 60,885. Sedangkan untuk jadwal *late acceptance*, jadwal ini adalah jadwal yang tidak hanya *feasible* tetapi juga sudah dioptimalkan. Sehingga nilai *proximity cost* yang didapatkan dari jadwal ini bisa turun ke angka 38,882. Perubahan nilai *proximity cost* dari jadwal *greedy* ke jadwal *late acceptance* dinilai cukup signifikan, yaitu sebesar 22,003. Tabel 6.7 menunjukkan jadwal hasil uji coba keempat dengan nilai *proximity cost* terkecil.

Tabel 6.7. Hasil Uji Coba 4

Kode Mata Kuliah	Sesi ke-
0001	7
0002	6
0003	10
0004	13
0005	1
0006	8
0007	14
0008	5
0009	3
0010	5
0011	4
0012	11

Kode Mata Kuliah	Sesi ke-
0013	2
0014	7
0015	14
0016	1
0017	5
0018	12
0019	10
0020	3
0021	9
0022	15
0023	7
0024	13
0025	1
0026	8
0027	6
0028	4
0029	2
0030	11
0031	4
0032	1

6.5.5. Skenario Uji Coba 5

Skenario uji coba yang kelima dilakukan dengan menjalankan *method move* dan *swap* secara acak. Dalam uji coba keempat, jumlah antrian diubah menjadi 100. Sedangkan jumlah iterasi dilakukan sebanyak 500.000 kali.

Untuk mencari nilai *proximity cost* yang terbaik, dilakukan 11 kali uji coba. Dalam 11 kali percobaan, nilai *proximity cost* minimum yang berhasil didapatkan yaitu sebesar 39,843 dan nilai *proximity cost* maksimum yang berhasil didapatkan yaitu sebesar 51,810. Sedangkan rata-ratanya sebesar 46,466.

Selanjutnya, nilai *proximity cost* yang didapat dari penerapan algoritma *late acceptance* tersebut dapat dibandingkan dengan nilai *proximity cost* dari jadwal yang dibuat secara manual dan jadwal yang diperoleh dari penerapan algoritma *greedy*. Perlu diingat bahwa jadwal manual adalah jadwal yang tidak *feasible* sehingga nilai *proximity cost*nya pun sangat besar, yaitu senilai 208.712. Di samping itu, jadwal *greedy* adalah jadwal yang sudah *feasible* tetapi belum optimal. Sehingga nilai *proximity cost* untuk jadwal *greedy* juga masih cukup besar, yaitu senilai 60,885. Sedangkan untuk jadwal *late acceptance*, jadwal ini adalah jadwal yang tidak hanya *feasible* tetapi juga sudah dioptimalkan. Sehingga nilai *proximity cost* yang didapatkan dari jadwal ini bisa turun ke angka 39,843. Perubahan nilai *proximity cost* dari jadwal *greedy* ke jadwal *late acceptance* dinilai cukup signifikan, yaitu sebesar 21,042. Tabel 6.8 menunjukkan jadwal hasil uji coba kelima dengan nilai *proximity cost* terkecil.

Tabel 6.8. Hasil Uji Coba 5

Kode Mata Kuliah	Sesi ke-
0001	4
0002	1
0003	14
0004	11
0005	5
0006	9
0007	2
0008	6
0009	8
0010	2
0011	12
0012	2
0013	7
0014	10

Kode Mata Kuliah	Sesi ke-
0015	4
0016	11
0017	15
0018	9
0019	14
0020	3
0021	8
0022	6
0023	10
0024	4
0025	1
0026	11
0027	5
0028	7
0029	12
0030	13
0031	12
0032	1

6.5.6. Skenario Uji Coba 6

Skenario uji coba yang keenam dilakukan dengan menjalankan *method move* dan *swap* secara acak. Dalam uji coba keenam, jumlah antrian diubah menjadi 50. Sedangkan jumlah iterasi dilakukan sebanyak 1.000.000 kali.

Untuk mencari nilai *proximity cost* yang terbaik, dilakukan 11 kali uji coba. Dalam 11 kali percobaan, nilai *proximity cost* minimum yang berhasil didapatkan yaitu sebesar 37,935 dan nilai *proximity cost* maksimum yang berhasil didapatkan yaitu sebesar 44,635. Sedangkan rata-ratanya sebesar 41,277.

Selanjutnya, nilai *proximity cost* yang didapat dari penerapan algoritma *late acceptance* tersebut dapat dibandingkan dengan nilai *proximity cost* dari jadwal yang dibuat secara manual dan

jadwal yang diperoleh dari penerapan algoritma *greedy*. Perlu diingat bahwa jadwal manual adalah jadwal yang tidak *feasible* sehingga nilai *proximity cost*nya pun sangat besar, yaitu senilai 208.712. Di samping itu, jadwal *greedy* adalah jadwal yang sudah *feasible* tetapi belum optimal. Sehingga nilai *proximity cost* untuk jadwal *greedy* juga masih cukup besar, yaitu senilai 60,885. Sedangkan untuk jadwal *late acceptance*, jadwal ini adalah jadwal yang tidak hanya *feasible* tetapi juga sudah dioptimalkan. Sehingga nilai *proximity cost* yang didapatkan dari jadwal ini bisa turun ke angka 37,935. Perubahan nilai *proximity cost* dari jadwal *greedy* ke jadwal *late acceptance* dinilai cukup signifikan, yaitu sebesar 22,95. Tabel 6.9 menunjukkan jadwal hasil uji coba keenam dengan nilai *proximity cost* terkecil.

Tabel 6.9. Hasil Uji Coba 6

Kode Mata Kuliah	Sesi ke-
0001	6
0002	9
0003	4
0004	7
0005	13
0006	12
0007	1
0008	11
0009	2
0010	4
0011	14
0012	1
0013	6
0014	10
0015	3
0016	13

Kode Mata Kuliah	Sesi ke-
0017	15
0018	8
0019	7
0020	2
0021	5
0022	11
0023	3
0024	4
0025	13
0026	9
0027	3
0028	4
0029	10
0030	14
0031	10
0032	13

6.5.7. Skenario Uji Coba 7

Skenario uji coba yang ketujuh dilakukan dengan menjalankan *method move* dan *swap* secara acak. Dalam uji coba ketujuh, jumlah antrian diubah menjadi 50. Sedangkan jumlah iterasi dilakukan sebanyak 500.000 kali.

Untuk mencari nilai *proximity cost* yang terbaik, dilakukan 11 kali uji coba. Dalam 11 kali percobaan, nilai *proximity cost* minimum yang berhasil didapatkan yaitu sebesar 38,644 dan nilai *proximity cost* maksimum yang berhasil didapatkan yaitu sebesar 45,160. Sedangkan rata-ratanya sebesar 42,470.

Selanjutnya, nilai *proximity cost* yang didapat dari penerapan algoritma *late acceptance* tersebut dapat dibandingkan dengan nilai *proximity cost* dari jadwal yang dibuat secara manual dan jadwal yang diperoleh dari penerapan algoritma *greedy*. Perlu diingat bahwa jadwal manual adalah jadwal yang tidak *feasible* sehingga nilai *proximity cost*nya pun sangat besar, yaitu senilai

208.712. Di samping itu, jadwal *greedy* adalah jadwal yang sudah *feasible* tetapi belum optimal. Sehingga nilai *proximity cost* untuk jadwal *greedy* juga masih cukup besar, yaitu senilai 60,885. Sedangkan untuk jadwal *late acceptance*, jadwal ini adalah jadwal yang tidak hanya *feasible* tetapi juga sudah dioptimalkan. Sehingga nilai *proximity cost* yang didapatkan dari jadwal ini bisa turun ke angka 38,644. Perubahan nilai *proximity cost* dari jadwal *greedy* ke jadwal *late acceptance* dinilai cukup signifikan, yaitu sebesar 22,241. Tabel 6.10 menunjukkan jadwal hasil uji coba ketujuh dengan nilai *proximity cost* terkecil.

Tabel 6.10. Hasil Uji Coba 7

Kode Mata Kuliah	Sesi ke-
0001	4
0002	12
0003	10
0004	7
0005	2
0006	5
0007	3
0008	1
0009	3
0010	13
0011	14
0012	9
0013	4
0014	6
0015	11
0016	10
0017	1
0018	5
0019	2

Kode Mata Kuliah	Sesi ke-
0020	8
0021	15
0022	13
0023	7
0024	11
0025	10
0026	12
0027	7
0028	11
0029	6
0030	9
0031	14
0032	10

6.5.8. Skenario Uji Coba 8

Skenario uji coba yang kedelapan dilakukan dengan menjalankan *method move* dan *swap* secara acak. Dalam uji coba kedelapan, jumlah antrian diubah menjadi 30. Sedangkan jumlah iterasi dilakukan sebanyak 1.000.000 kali.

Untuk mencari nilai *proximity cost* yang terbaik, dilakukan 11 kali uji coba. Dalam 11 kali percobaan, nilai *proximity cost* minimum yang berhasil didapatkan yaitu sebesar 37,795 dan nilai *proximity cost* maksimum yang berhasil didapatkan yaitu sebesar 47,028. Sedangkan rata-ratanya sebesar 40,083.

Selanjutnya, nilai *proximity cost* yang didapat dari penerapan algoritma *late acceptance* tersebut dapat dibandingkan dengan nilai *proximity cost* dari jadwal yang dibuat secara manual dan jadwal yang diperoleh dari penerapan algoritma *greedy*. Perlu diingat bahwa jadwal manual adalah jadwal yang tidak *feasible* sehingga nilai *proximity cost*nya pun sangat besar, yaitu senilai 208.712. Di samping itu, jadwal *greedy* adalah jadwal yang sudah *feasible* tetapi belum optimal. Sehingga nilai *proximity cost* untuk jadwal *greedy* juga masih cukup besar, yaitu senilai

60,885. Sedangkan untuk jadwal *late acceptance*, jadwal ini adalah jadwal yang tidak hanya *feasible* tetapi juga sudah dioptimalkan. Sehingga nilai *proximity cost* yang didapatkan dari jadwal ini bisa turun ke angka 37,795. Perubahan nilai *proximity cost* dari jadwal *greedy* ke jadwal *late acceptance* dinilai cukup signifikan, yaitu sebesar 23,09. Tabel 6.11 menunjukkan jadwal hasil uji coba kedelapan dengan nilai *proximity cost* terkecil.

Tabel 6.11. Hasil Uji Coba 8

Kode Mata Kuliah	Sesi ke-
0001	4
0002	12
0003	9
0004	3
0005	6
0006	2
0007	13
0008	7
0009	13
0010	4
0011	11
0012	14
0013	8
0014	5
0015	1
0016	3
0017	4
0018	2
0019	9
0020	15
0021	10

Kode Mata Kuliah	Sesi ke-
0022	7
0023	6
0024	14
0025	1
0026	3
0027	6
0028	14
0029	11
0030	12
0031	5
0032	1

6.5.9. Skenario Uji Coba 9

Skenario uji coba yang kesembilan dilakukan dengan menjalankan *method move* dan *swap* secara acak. Dalam uji coba kesembilan, jumlah antrian diubah menjadi 30. Sedangkan jumlah iterasi dilakukan sebanyak 500.000 kali.

Untuk mencari nilai *proximity cost* yang terbaik, dilakukan 11 kali uji coba. Dalam 11 kali percobaan, nilai *proximity cost* minimum yang berhasil didapatkan yaitu sebesar 36,083 dan nilai *proximity cost* maksimum yang berhasil didapatkan yaitu sebesar 44,473. Sedangkan rata-ratanya sebesar 41,062.

Selanjutnya, nilai *proximity cost* yang didapat dari penerapan algoritma *late acceptance* tersebut dapat dibandingkan dengan nilai *proximity cost* dari jadwal yang dibuat secara manual dan jadwal yang diperoleh dari penerapan algoritma *greedy*. Perlu diingat bahwa jadwal manual adalah jadwal yang tidak *feasible* sehingga nilai *proximity cost*nya pun sangat besar, yaitu senilai 208.712. Di samping itu, jadwal *greedy* adalah jadwal yang sudah *feasible* tetapi belum optimal. Sehingga nilai *proximity cost* untuk jadwal *greedy* juga masih cukup besar, yaitu senilai 60,885. Sedangkan untuk jadwal *late acceptance*, jadwal ini adalah jadwal yang tidak hanya *feasible* tetapi juga sudah dioptimalkan. Sehingga nilai *proximity cost* yang didapatkan

dari jadwal ini bisa turun ke angka 36,083. Perubahan nilai *proximity cost* dari jadwal *greedy* ke jadwal *late acceptance* dinilai cukup signifikan, yaitu sebesar 24,802. Tabel 6.12 menunjukkan jadwal hasil uji coba kesembilan dengan nilai *proximity cost* terkecil.

Tabel 6.12. Hasil Uji Coba 9

Kode Mata Kuliah	Sesi ke-
0001	7
0002	3
0003	14
0004	5
0005	11
0006	8
0007	10
0008	2
0009	6
0010	3
0011	14
0012	12
0013	9
0014	7
0015	1
0016	5
0017	15
0018	2
0019	3
0020	6
0021	8
0022	13
0023	1
0024	12

Kode Mata Kuliah	Sesi ke-
0025	5
0026	4
0027	1
0028	7
0029	4
0030	11
0031	7
0032	5

6.5.10. Skenario Uji Coba 10

Skenario uji coba yang kesepuluh dilakukan dengan menjalankan *method move* dan *swap* secara acak. Dalam uji coba kesepuluh, jumlah antrian diubah menjadi 10. Sedangkan jumlah iterasi dilakukan sebanyak 1.000.000 kali.

Untuk mencari nilai *proximity cost* yang terbaik, dilakukan 11 kali uji coba. Dalam 11 kali percobaan, nilai *proximity cost* minimum yang berhasil didapatkan yaitu sebesar 37,795 dan nilai *proximity cost* maksimum yang berhasil didapatkan yaitu sebesar 44,735. Sedangkan rata-ratanya sebesar 40,824.

Selanjutnya, nilai *proximity cost* yang didapat dari penerapan algoritma *late acceptance* tersebut dapat dibandingkan dengan nilai *proximity cost* dari jadwal yang dibuat secara manual dan jadwal yang diperoleh dari penerapan algoritma *greedy*. Perlu diingat bahwa jadwal manual adalah jadwal yang tidak *feasible* sehingga nilai *proximity cost*nya pun sangat besar, yaitu senilai 208.712. Di samping itu, jadwal *greedy* adalah jadwal yang sudah *feasible* tetapi belum optimal. Sehingga nilai *proximity cost* untuk jadwal *greedy* juga masih cukup besar, yaitu senilai 60,885. Sedangkan untuk jadwal *late acceptance*, jadwal ini adalah jadwal yang tidak hanya *feasible* tetapi juga sudah dioptimalkan. Sehingga nilai *proximity cost* yang didapatkan dari jadwal ini bisa turun ke angka 37,795. Perubahan nilai *proximity cost* dari jadwal *greedy* ke jadwal *late acceptance* dinilai cukup signifikan, yaitu sebesar 23,09. Tabel 6.13

menunjukkan jadwal hasil uji coba kesepuluh dengan nilai *proximity cost* terkecil.

Tabel 6.13. Hasil Uji Coba 10

Kode Mata Kuliah	Sesi ke-
0001	11
0002	14
0003	9
0004	6
0005	3
0006	10
0007	5
0008	12
0009	1
0010	3
0011	11
0012	5
0013	2
0014	8
0015	13
0016	3
0017	10
0018	12
0019	4
0020	15
0021	1
0022	7
0023	6
0024	13
0025	9
0026	3

Kode Mata Kuliah	Sesi ke-
0027	6
0028	13
0029	3
0030	8
0031	11
0032	14

6.5.11. Skenario Uji Coba 11

Skenario uji coba yang kesebelas dilakukan dengan menjalankan *method move* dan *swap* secara acak. Dalam uji coba kesebelas, jumlah antrian diubah menjadi 10. Sedangkan jumlah iterasi dilakukan sebanyak 500.000 kali.

Untuk mencari nilai *proximity cost* yang terbaik, dilakukan 11 kali uji coba. Dalam 11 kali percobaan, nilai *proximity cost* minimum yang berhasil didapatkan yaitu sebesar 36,503 dan nilai *proximity cost* maksimum yang berhasil didapatkan yaitu sebesar 45,824. Sedangkan rata-ratanya sebesar 41,686.

Selanjutnya, nilai *proximity cost* yang didapat dari penerapan algoritma *late acceptance* tersebut dapat dibandingkan dengan nilai *proximity cost* dari jadwal yang dibuat secara manual dan jadwal yang diperoleh dari penerapan algoritma *greedy*. Perlu diingat bahwa jadwal manual adalah jadwal yang tidak *feasible* sehingga nilai *proximity cost*nya pun sangat besar, yaitu senilai 208.712. Di samping itu, jadwal *greedy* adalah jadwal yang sudah *feasible* tetapi belum optimal. Sehingga nilai *proximity cost* untuk jadwal *greedy* juga masih cukup besar, yaitu senilai 60,885. Sedangkan untuk jadwal *late acceptance*, jadwal ini adalah jadwal yang tidak hanya *feasible* tetapi juga sudah dioptimalkan. Sehingga nilai *proximity cost* yang didapatkan dari jadwal ini bisa turun ke angka 36,503. Perubahan nilai *proximity cost* dari jadwal *greedy* ke jadwal *late acceptance* dinilai cukup signifikan, yaitu sebesar 24,382. Tabel 6.14 menunjukkan jadwal hasil uji coba kesebelas dengan nilai *proximity cost* terkecil.

Tabel 6.14. Hasil Uji Coba 11

Kode Mata Kuliah	Sesi ke-
0001	6
0002	8
0003	11
0004	13
0005	3
0006	1
0007	4
0008	9
0009	6
0010	4
0011	7
0012	14
0013	2
0014	10
0015	5
0016	13
0017	1
0018	9
0019	11
0020	12
0021	6
0022	15
0023	8
0024	5
0025	13
0026	14
0027	8
0028	13
0029	2

Kode Mata Kuliah	Sesi ke-
0030	3
0031	10
0032	13

6.6. Perbandingan Hasil Algoritma *Late acceptance*

Setelah mengetahui hasil uji coba dengan skenario yang berbeda-beda, maka perlu dilakukan analisis lebih lanjut mengenai perbandingan hasil dari uji coba satu dengan yang lainnya. Penjelasannya sebagai berikut.

6.6.1. Perbandingan Hasil Uji Coba 1, 2 dan 3

Hasil yang akan dibandingkan adalah uji coba skenario 1, 2 dan 3. Hal ini dikarenakan ketiga skenario uji coba tersebut menggunakan jumlah antrian dan jumlah iterasi yang sama. Pada ketiga uji coba ini, jumlah antrian yang digunakan sebanyak 5.000 dan jumlah iterasi yang digunakan sebanyak 1.000.000. Parameter yang digunakan pada ketiga uji coba ini dipilih berdasarkan referensi dari penelitian sebelumnya. Perbedaan yang terdapat pada ketiga uji coba ini adalah berdasarkan *method* yang digunakan. Uji coba pertama menggunakan *method move* dan *swap* secara acak, uji coba kedua menggunakan *method move* saja, dan uji coba ketiga menggunakan *method swap* saja. Tabel 6.15 menunjukkan perbedaan skenario pada ketiga uji coba yang dijalankan.

Tabel 6.15. Perbandingan Parameter Uji Coba 1, 2, dan 3

Skenario	Antrian	Iterasi	*Method*
Uji Coba 1	5000	1.000.000	*Random (Move & Swap)*
Uji Coba 2	5000	1.000.000	*Move*
Uji Coba 3	5000	1.000.000	*Swap*

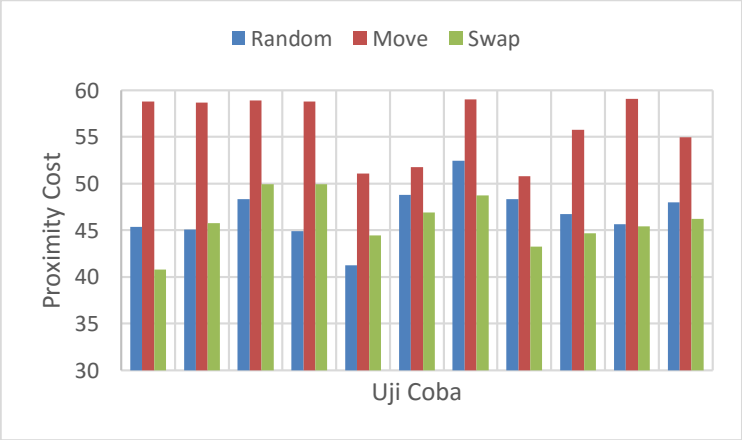
Ketiga skenario uji coba tersebut dijalankan sebanyak 11 kali. Dari percobaan sebanyak 11 kali, didapatkan nilai *proximity cost* seperti yang ditampilkan pada tabel 6.16. Dapat dilihat pada tabel 6.16 bahwa nilai *proximity cost* yang paling baik diperoleh dari hasil uji coba 3. Hal ini dikarenakan uji coba 3

menunjukkan nilai *proximity cost* minimum, nilai *proximity cost* maksimum, dan nilai rata-rata *proximity cost* yang lebih rendah jika dibandingkan dengan nilai *proximity cost* dari uji coba 1 dan uji coba 2.

Tabel 6.16. Perbandingan Hasil Uji Coba 1, 2, dan 3

Perbandingan	UC 1	UC 2	UC 3
Proximity Cost Minimum	41.252	50.792	40.801
Proximity Cost Maksimum	52.457	59.111	49.910
Rata-Rata Proximity Cost	46.806	56.157	46.004

Gambar 6.3 menunjukkan diagram batang dari perbandingan hasil uji coba 1, 2, dan 3 yang dijalankan sebanyak 11 kali. Dari grafik ini terlihat bahwa hasil uji coba 2 mempunyai nilai *proximity cost* yang selalu lebih besar dibandingkan dengan hasil uji coba 1 dan 3.



Gambar 6.3. Perbandingan Hasil Uji Coba 1, 2, dan 3

Gambar 6.4 menunjukkan *box plot* dari perbandingan hasil uji coba 1, 2, dan 3. Dapat dilihat bahwa *range* nilai uji coba 2 berada di kisaran yang lebih tinggi dari *range* nilai uji coba 1 dan 3. Nilai uji coba 2 juga mempunyai jangkauan yang lebih lebar, hal ini menunjukkan bahwa hasil uji coba 2 mempunyai nilai yang sangat variatif. Hasil uji coba 1 mempunyai *range*

nilai yang paling sempit, akan tetapi uji coba 1 tidak menghasilkan nilai *proximity cost* terendah. Nilai *proximity cost* yang paling rendah ditunjukkan oleh hasil uji coba 3. Akan tetapi, selisih nilai *proximity cost* antara uji coba 1 dan 3 relatif kecil. Sehingga dapat disimpulkan bahwa uji coba 1 dan 3 dapat menghasilkan nilai *proximity cost* yang hampir sama.

Gambar 6.4. *Box Plot* Perbandingan Hasil Uji Coba 1, 2, dan 3

Nilai *proximity cost* maksimum yang didapat dari grafik tersebut adalah 59,111 yang merupakan hasil uji coba 2. Sedangkan nilai *proximity cost* minimum yang didapat dari grafik tersebut adalah 40,801 yang merupakan hasil dari uji coba 3. Dari grafik ini dapat disimpulkan bahwa uji coba 3 dengan menggunakan methode *swap* menunjukkan nilai *proximity cost* yang lebih rendah, sehingga jadwal yang dihasilkan pun menjadi lebih baik.

6.6.2. Perbandingan Hasil Uji Coba 4, 6, 8, dan 10

Hasil yang akan dibandingkan adalah uji coba skenario 4, 6, 8, dan 10. Hal ini dikarenakan keempat skenario uji coba tersebut menggunakan jumlah iterasi yang sama, yaitu sebanyak 1.000.000 kali iterasi. Pada keempat uji coba ini, jumlah antrian yang digunakan berbeda-beda, yaitu sebanyak 100, 50, 30, dan

10. Tabel 6.17 menunjukkan perbedaan skenario pada keempat uji coba yang dijalankan.

Tabel 6.17. Perbandingan Parameter Uji Coba 4, 6, 8, dan 10

Skenario	Antrian	Iterasi	*Method*
Uji Coba 4	100	1.000.000	*Random (Move & Swap)*
Uji Coba 6	50	1.000.000	*Random (Move & Swap)*
Uji Coba 8	30	1.000.000	*Random (Move & Swap)*
Uji Coba 10	10	1.000.000	*Random (Move & Swap)*

Keempat skenario uji coba tersebut dijalankan sebanyak 11 kali. Dari percobaan sebanyak 11 kali, didapatkan nilai *proximity cost* seperti yang ditampilkan pada tabel 6.18. Dapat dilihat pada tabel 6.18 bahwa nilai *proximity cost* yang paling baik diperoleh dari hasil uji coba 10. Hal ini dikarenakan uji coba 10 menunjukkan nilai *proximity cost* minimum, nilai *proximity cost* maksimum, dan nilai rata-rata *proximity cost* yang lebih rendah jika dibandingkan dengan nilai *proximity cost* dari uji coba 4, 6, dan uji coba 8.

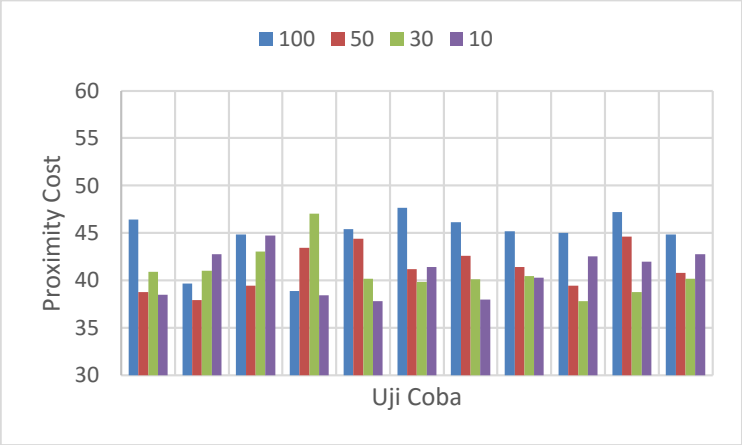
Tabel 6.18. Perbandingan Hasil Uji Coba 4, 6, 8, dan 10

Perbandingan	UC 4	UC 6	UC 8	UC 10
Proximity Cost Minimum	38.882	37.935	37.795	37.795
Proximity Cost Maksimum	47.660	44.635	47.028	44.735
Rata-Rata Proximity Cost	44.654	41.277	40.843	40.824

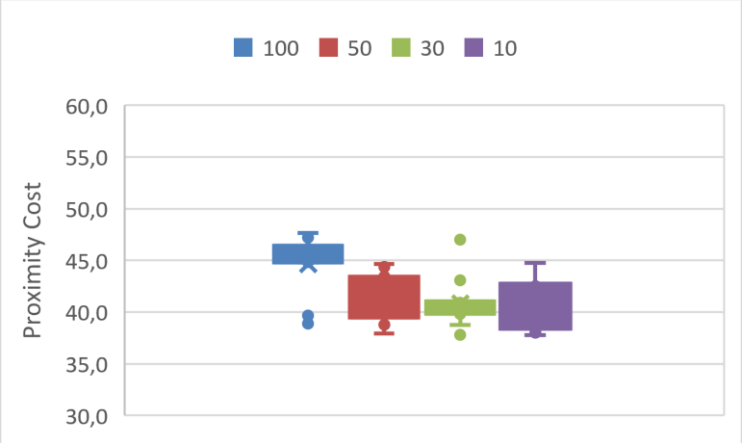
Gambar 6.5 menunjukkan diagram batang dari perbandingan hasil uji coba 4, 6, 8, dan 10 yang dijalankan sebanyak 11 kali. Dari gambar 6.5 terlihat bahwa hasil uji coba 4 mempunyai nilai *proximity cost* yang selalu lebih besar dibandingkan dengan hasil uji coba 6, 8, dan 10.

Sedangkan gambar 6.6 menunjukkan *box plot* dari perbandingan hasil uji coba 4, 6, 8, dan 10. Dapat dilihat bahwa *range* nilai uji coba 4 berada di kisaran yang lebih tinggi dari *range* nilai uji coba 6, 8, dan 10. Uji coba 6 dan 10 menunjukkan jangkauan nilai yang lebih lebar dibandingkan dengan uji coba 4 dan 8. Uji coba 8 mempunyai *range* nilai paling sempit. Hal ini menunjukkan bahwa hasil uji coba 8 mempunyai nilai

proximity cost yang hampir sama. Akan tetapi, nilai *proximity cost* terendah berhasil ditunjukkan oleh uji coba 10.



Gambar 6.5. Perbandingan Hasil Uji Coba 4, 6, 8, 10



Gambar 6.6. *Box Plot* Perbandingan Hasil Uji Coba 4, 6, 8, 10

Nilai *proximity cost* maksimum yang didapat dari grafik tersebut adalah 47,660 yang merupakan hasil uji coba 4. Sedangkan nilai *proximity cost* minimum yang didapat dari grafik tersebut adalah 37,795 yang merupakan hasil dari uji

coba 10. Dari grafik ini dapat disimpulkan bahwa uji coba 10 dengan jumlah antrian 10 menunjukkan nilai *proximity cost* yang lebih rendah, sehingga jadwal yang dihasilkan pun menjadi lebih baik.

6.6.3. Perbandingan Hasil Uji Coba 5, 7, 9, dan 11

Hasil yang akan dibandingkan adalah uji coba skenario 5, 7, 9, dan 11. Hal ini dikarenakan keempat skenario uji coba tersebut menggunakan jumlah iterasi yang sama yaitu sebanyak 500.000 kali iterasi. Pada keempat uji coba ini, jumlah antrian yang digunakan berbeda-beda, yaitu sebanyak 100, 50, 30, dan 10. Tabel 6.19 menunjukkan perbedaan skenario pada keempat uji coba yang dijalankan.

Tabel 6.19. Perbandingan Parameter Uji Coba 5, 7, 9, dan 11

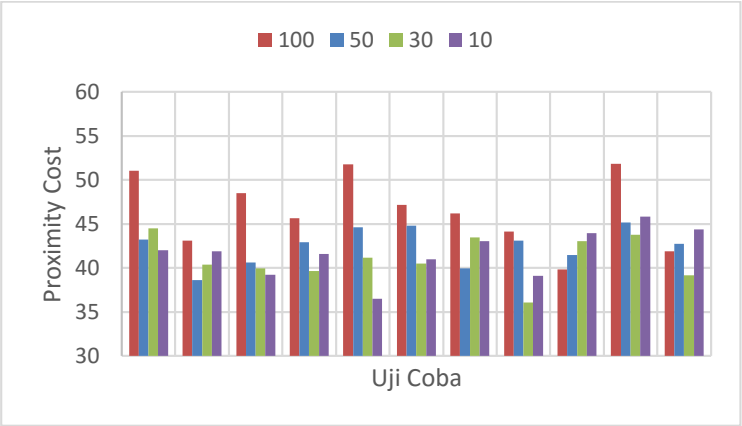
Skenario	Antrian	Iterasi	*Method*
Uji Coba 5	100	500.000	*Random (Move & Swap)*
Uji Coba 7	50	500.000	*Random (Move & Swap)*
Uji Coba 9	30	500.000	*Random (Move & Swap)*
Uji Coba 11	10	500.000	*Random (Move & Swap)*

Keempat skenario uji coba tersebut dijalankan sebanyak 11 kali. Dari percobaan sebanyak 11 kali, didapatkan nilai *proximity cost* seperti yang ditampilkan pada tabel 6.20. Dapat dilihat pada tabel 6.20 bahwa nilai *proximity cost* yang paling baik diperoleh dari hasil uji coba 9. Hal ini dikarenakan uji coba 9 menunjukkan nilai *proximity cost* minimum, nilai *proximity cost* maksimum, dan nilai rata-rata *proximity cost* yang lebih rendah jika dibandingkan dengan nilai *proximity cost* dari uji coba 5, 7, dan uji coba 11.

Tabel 6.20. Perbandingan Hasil Uji Coba 5, 7, 9, dan 11

Perbandingan	UC 5	UC 7	UC 9	UC 11
Proximity Cost Minimum	39.843	38.644	36.083	36.503
Proximity Cost Maksimum	51.810	45.160	44.473	45.824
Rata-Rata Proximity Cost	46.466	42.470	41.062	41.686

Gambar 6.7 menunjukkan diagram batang dari perbandingan hasil uji coba 5, 7, 9, dan 11 yang dijalankan sebanyak 11 kali. Dari gambar 6.7 terlihat bahwa hasil uji coba 5 mempunyai nilai *proximity cost* yang selalu lebih besar dibandingkan dengan hasil uji coba 7, 9, dan 11.

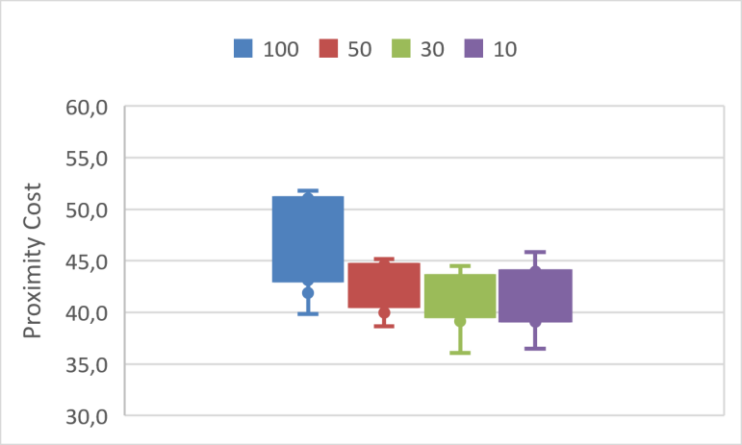


Gambar 6.7. Perbandingan Hasil Uji Coba 5, 7, 9, dan 11

Sedangkan gambar 6.8 menunjukkan *box plot* dari perbandingan hasil uji coba 5, 7, 9, dan 11. Dapat dilihat bahwa *range* nilai uji coba 5 berada di kisaran yang lebih tinggi dari *range* nilai uji coba 7, 9, dan 11. Uji coba 5 juga menunjukkan jangkauan nilai yang paling lebar jika dibandingkan dengan uji coba 7, 9, dan 11. Uji coba 9 dan 11 mempunyai jangkauan nilai *proximity cost* yang hampir sama. Uji coba 11 mempunyai *range* yang sedikit lebih lebar dibandingkan dengan uji coba 9. Akan tetapi, nilai *proximity cost* terendah berhasil ditunjukkan oleh uji coba 9.

Nilai *proximity cost* maksimum yang didapat dari grafik tersebut adalah 51,810 yang merupakan hasil uji coba 5. Sedangkan nilai *proximity cost* minimum yang didapat dari grafik tersebut adalah 36,083 yang merupakan hasil dari uji coba 9. Dari grafik ini dapat disimpulkan bahwa uji coba 9 dengan jumlah antrian 30 menunjukkan nilai *proximity cost*

yang lebih rendah, sehingga jadwal yang dihasilkan pun menjadi lebih baik.



Gambar 6.8. *Box plot* Perbandingan Hasil Uji Coba 5, 7, 9, dan 11

6.6.4. Perbandingan Hasil Uji Coba 4 dan 5

Hasil yang akan dibandingkan adalah uji coba skenario 4 dan 5. Hal ini dikarenakan kedua skenario uji coba tersebut menggunakan jumlah antrian yang sama yaitu sebanyak 100. Pada kedua uji coba ini, digunakan jumlah iterasi yang berbeda, yaitu sebanyak 1.000.000 dan 500.000. Tabel 6.21 menunjukkan perbedaan skenario pada kedua uji coba yang dijalankan.

Tabel 6.21. Perbandingan Parameter Uji Coba 4 dan 5

Skenario	Antrian	Iterasi	*Method*
Uji Coba 4	100	1.000.000	*Random (Move & Swap)*
Uji Coba 5	100	500.000	*Random (Move & Swap)*

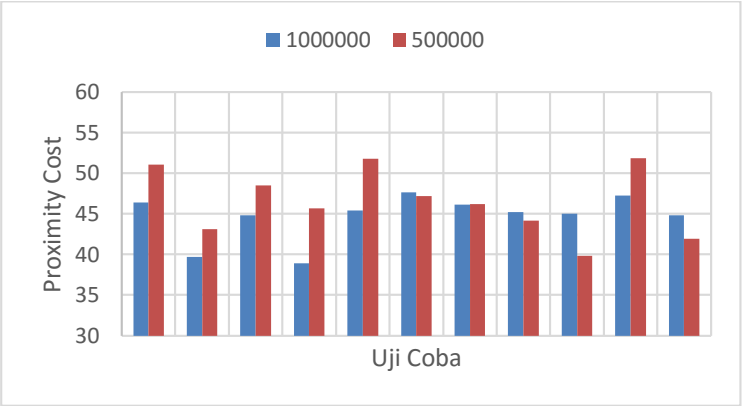
Kedua skenario uji coba tersebut dijalankan sebanyak 11 kali. Dari percobaan sebanyak 11 kali, didapatkan nilai *proximity cost* seperti yang ditampilkan pada tabel 6.22. Dapat dilihat pada tabel 6.22 bahwa nilai *proximity cost* yang paling baik diperoleh dari hasil uji coba 4. Hal ini dikarenakan uji coba 4

menunjukkan nilai *proximity cost* minimum, nilai *proximity cost* maksimum, dan nilai rata-rata *proximity cost* yang lebih rendah jika dibandingkan dengan nilai *proximity cost* dari uji coba 5.

Tabel 6.22. Perbandingan Hasil Uji Coba 4 dan 5

Perbandingan	UC 4	UC 5
Proximity Cost Minimum	38.882	39.843
Proximity Cost Maksimum	47.660	51.810
Rata-Rata Proximity Cost	44.654	46.466

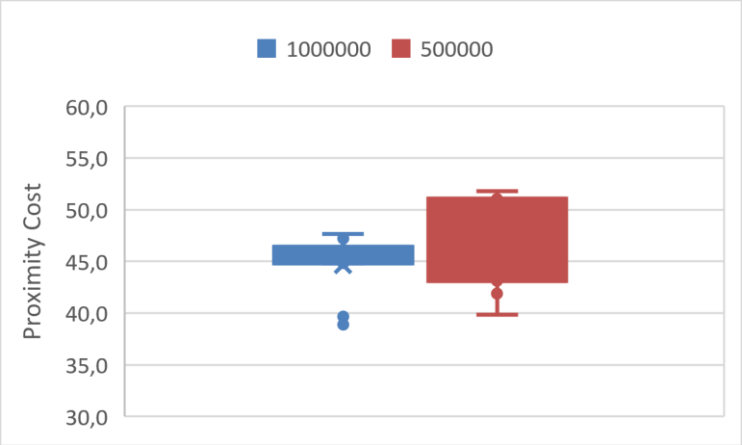
Gambar 6.9 menunjukkan diagram batang dari perbandingan hasil uji coba 4 dan 5 yang dijalankan sebanyak 11 kali. Dari gambar 6.10 terlihat bahwa hasil uji coba 5 mempunyai nilai *proximity cost* yang relatif lebih besar dibandingkan dengan hasil uji coba 4. Namun, pada beberapa iterasi, dapat dijumpai hasil uji coba 4 yang menunjukkan nilai *proximity cost* yang lebih besar dibandingkan hasil uji coba 5.



Gambar 6.9. Perbandingan Hasil Uji Coba 4 dan 5

Sedangkan gambar 6.10 menunjukkan *box plot* dari perbandingan hasil uji coba 4 dan 5. Dapat dilihat bahwa hasil uji coba 5 mempunyai jangkauan nilai yang lebih luas dibandingkan hasil uji coba 4. Pada hasil uji coba 5 juga tidak terdeteksi adanya nilai yang menjadi *outliers*, berbeda dengan

hasil uji coba 4 yang mempunyai beberapa nilai *outliers.* Akan tetapi, nilai *proximity cost* terendah berhasil ditunjukkan oleh uji coba 4.



Gambar 6.10. *Box Plot* Perbandingan Hasil Uji Coba 4 dan 5

Nilai *proximity cost* maksimum yang didapat dari grafik tersebut adalah 51,810 yang merupakan hasil uji coba 5. Sedangkan nilai *proximity cost* minimum yang didapat dari grafik tersebut adalah 38,882 yang merupakan hasil dari uji coba 4. Dari grafik ini dapat disimpulkan bahwa uji coba 4 dengan jumlah iterasi 1.000.000 menunjukkan nilai *proximity cost* yang lebih rendah, sehingga jadwal yang dihasilkan pun menjadi lebih baik.

6.6.5. Perbandingan Hasil Uji Coba 6 dan 7

Hasil yang akan dibandingkan adalah uji coba skenario 6 dan 7. Hal ini dikarenakan kedua skenario uji coba tersebut menggunakan jumlah antrian yang sama yaitu sebanyak 50. Pada kedua uji coba ini, digunakan jumlah iterasi yang berbeda, yaitu sebanyak 1.000.000 dan 500.000. Tabel 6.23 menunjukkan perbedaan skenario pada kedua uji coba yang dijalankan.

Tabel 6.23. Perbandingan Parameter Uji Coba 6 dan 7

Skenario	Antrian	Iterasi	*Method*
Uji Coba 6	50	1.000.000	*Random (Move & Swap)*
Uji Coba 7	50	500.000	*Random (Move & Swap)*

Kedua skenario uji coba tersebut dijalankan sebanyak 11 kali. Dari percobaan sebanyak 11 kali, didapatkan nilai *proximity cost* seperti yang ditampilkan pada tabel 6.24. Dapat dilihat pada tabel 6.24 bahwa nilai *proximity cost* yang paling baik diperoleh dari hasil uji coba 6. Hal ini dikarenakan uji coba 6 menunjukkan nilai *proximity cost* minimum, nilai *proximity cost* maksimum, dan nilai rata-rata *proximity cost* yang lebih rendah jika dibandingkan dengan nilai *proximity cost* dari uji coba 7.

Tabel 6.24. Perbandingan Hasil Uji Coba 6 dan 7

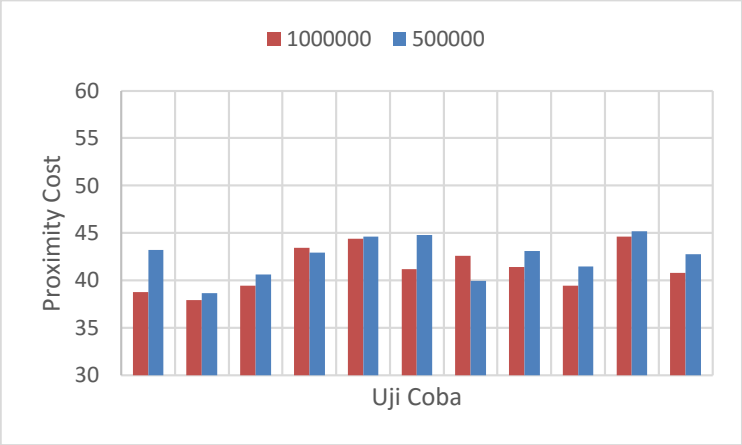
Perbandingan	UC 6	UC 7
Proximity Cost Minimum	37.935	38.644
Proximity Cost Maksimum	44.635	45.160
Rata-Rata Proximity Cost	41.277	42.470

Gambar 6.11 menunjukkan *diagram batang dari* perbandingan hasil uji coba 6 dan 7 yang dijalankan sebanyak 11 kali. Dari grafik tersebut, dapat dilihat bahwa hasil uji coba 7 menunjukkan nilai *proximity cost* yang lebih tinggi dibandingkan dengan hasil uji coba 6.

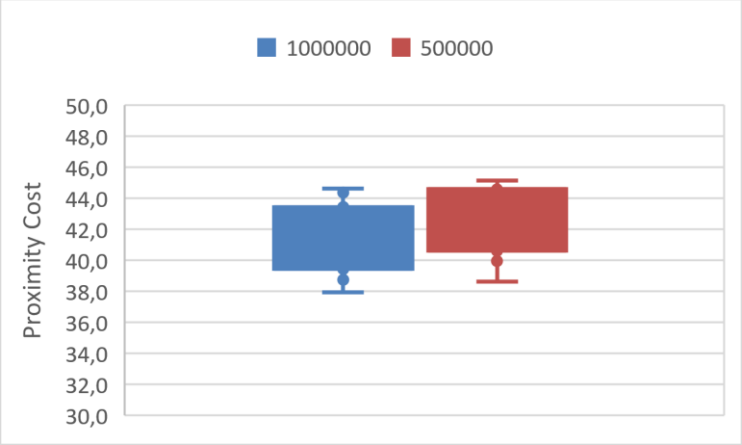
Sedangkan dari gambar 6.12, dapat dilihat bahwa uji coba 6 dan 7 mempunyai *range* nilai *proximity cost* yang hampir sama. Hanya saja, hasil uji coba 7 berada di kisaran yang lebih tinggi karena nilai *proximity cost* yang didapat lebih besar daripada hasil uji coba 6.

Nilai *proximity cost* maksimum yang didapat dari grafik tersebut adalah 45,160 yang merupakan hasil uji coba 7. Sedangkan nilai *proximity cost* minimum yang didapat dari grafik tersebut adalah 37,935 yang merupakan hasil dari uji coba 6. Dari grafik ini dapat disimpulkan bahwa uji coba 6 dengan jumlah iterasi 1.000.000 menunjukkan nilai *proximity*

cost yang lebih rendah, sehingga jadwal yang dihasilkan pun menjadi lebih baik.



Gambar 6.11. Perbandingan Hasil Uji Coba 6 dan 7



Gambar 6.12. *Box Plot* Perbandingan Hasil Uji Coba 6 dan 7

6.6.6. Perbandingan Hasil Uji Coba 8 dan 9

Hasil yang akan dibandingkan adalah uji coba skenario 8 dan 9. Hal ini dikarenakan kedua skenario uji coba tersebut

menggunakan jumlah antrian yang sama yaitu sebanyak 30. Pada kedua uji coba ini, digunakan jumlah iterasi yang berbeda, yaitu sebanyak 1.000.000 dan 500.000. Tabel 6.25 menunjukkan perbedaan skenario pada kedua uji coba yang dijalankan.

Tabel 6.25. Perbandingan Parameter Uji Coba 8 dan 9

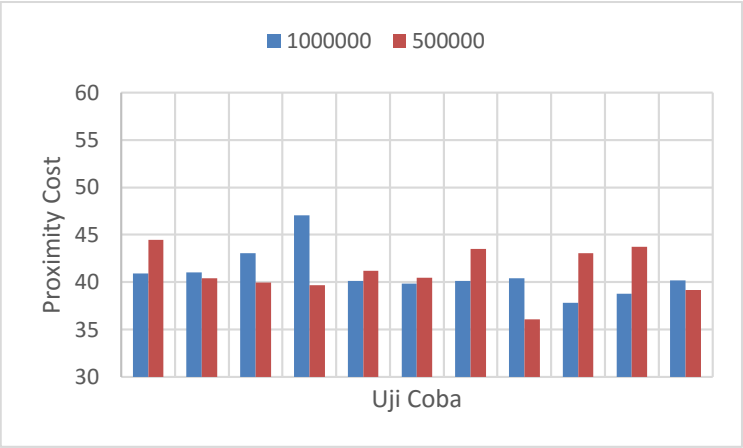
Skenario	Antrian	Iterasi	*Method*
Uji Coba 8	30	1.000.000	*Random (Move & Swap)*
Uji Coba 9	30	500.000	*Random (Move & Swap)*

Kedua skenario uji coba tersebut dijalankan sebanyak 11 kali. Dari percobaan sebanyak 11 kali, didapatkan nilai *proximity cost* seperti yang ditampilkan pada tabel 6.26. Dapat dilihat pada tabel 6.26 bahwa nilai *proximity cost* yang paling baik diperoleh dari hasil uji coba 9. Hal ini dikarenakan uji coba 9 menunjukkan nilai *proximity cost* minimum dan nilai *proximity cost* maksimum yang lebih rendah jika dibandingkan dengan nilai *proximity cost* dari uji coba 8.

Tabel 6.26. Perbandingan Hasil Uji Coba 8 dan 9

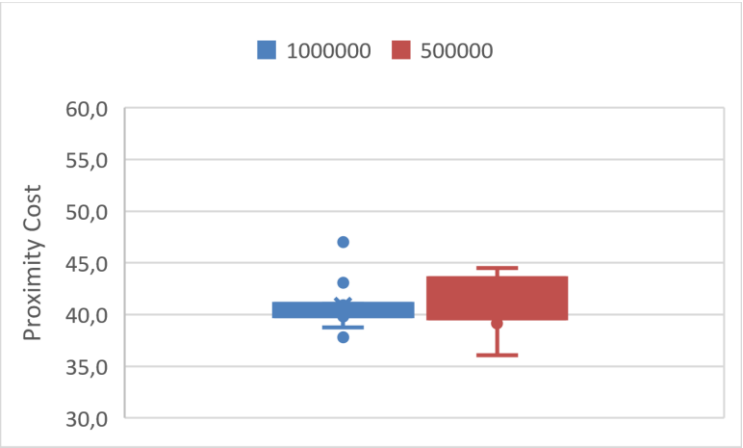
Perbandingan	UC 8	UC 9
Proximity Cost Minimum	37.795	36.083
Proximity Cost Maksimum	47.028	44.473
Rata-Rata Proximity Cost	40.843	41.062

Gambar 6.13 menunjukkan diagram batang dari perbandingan hasil uji coba 8 dan 9 yang dijalankan sebanyak 11 kali. Dari grafik tersebut, dapat dilihat bahwa hasil uji coba 8 dan 9 menunjukkan nilai *proximity cost* yang relatif sama. Pada beberapa iterasi, hasil uji coba 8 menunjukkan nilai *proximity cost* yang jauh lebih tinggi dibandingkan hasil uji coba 9. Akan tetapi, pada iterasi lain dapat ditemukan bahwa uji coba 9 juga dapat menunjukkan nilai *proximity cost* yang lebih tinggi dibandingkan uji coba 8.



Gambar 6.13. Perbandingan Hasil Uji Coba 8 dan 9

Sedangkan dari gambar 6.14, dapat dilihat bahwa uji coba 8 dan 9 mempunyai *range* nilai *proximity cost* yang hampir sama. Namun, uji coba 9 mempunyai jangkauan nilai *proximity cost* yang lebih luas dibandingkan dengan uji coba 8.



Gambar 6.14. *Box Plot* Perbandingan Hasil Uji Coba 8 dan 9

Nilai *proximity cost* maksimum yang didapat dari grafik tersebut adalah 47,028 yang merupakan hasil uji coba 8.

Sedangkan nilai *proximity cost* minimum yang didapat dari grafik tersebut adalah 36,083 yang merupakan hasil dari uji coba 9. Dari grafik ini dapat disimpulkan bahwa uji coba 9 dengan jumlah iterasi 500.000 menunjukkan nilai *proximity cost* yang paling rendah. Tetapi secara rata-rata, hasil uji coba 8 dengan jumlah iterasi 1.000.000 menunjukkan nilai *proximity cost* yang lebih baik, sehingga jadwal yang dihasilkan pun menjadi lebih baik.

6.6.7. Perbandingan Hasil Uji Coba 10 dan 11

Hasil yang akan dibandingkan adalah uji coba skenario 10 dan 11. Hal ini dikarenakan kedua skenario uji coba tersebut menggunakan jumlah antrian yang sama yaitu sebanyak 10. Pada kedua uji coba ini, digunakan jumlah iterasi yang berbeda, yaitu sebanyak 1.000.000 dan 500.000. Tabel 6.27 menunjukkan perbedaan skenario pada kedua uji coba yang dijalankan.

Tabel 6.27. Perbandingan Parameter Uji Coba 10 dan 11

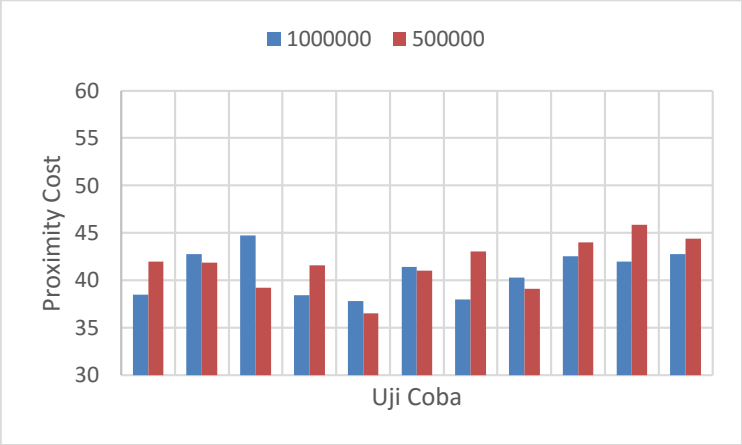
Skenario	Antrian	Iterasi	*Method*
Uji Coba 10	10	1.000.000	*Random (Move & Swap)*
Uji Coba 11	10	500.000	*Random (Move & Swap)*

Kedua skenario uji coba tersebut dijalankan sebanyak 11 kali. Dari percobaan sebanyak 11 kali, didapatkan nilai *proximity cost* seperti yang ditampilkan pada tabel 6.28. Dapat dilihat pada tabel 6.28 bahwa nilai *proximity cost* yang paling baik diperoleh dari hasil uji coba 11. Hal ini dikarenakan uji coba 11 menunjukkan nilai *proximity cost* minimum yang lebih rendah jika dibandingkan dengan nilai *proximity cost* dari uji coba 10. Namun secara rata-rata, uji coba 10 menunjukkan nilai *proximity cost* yang lebih baik.

Tabel 6.28. Perbandingan Hasil Uji Coba 10 dan 11

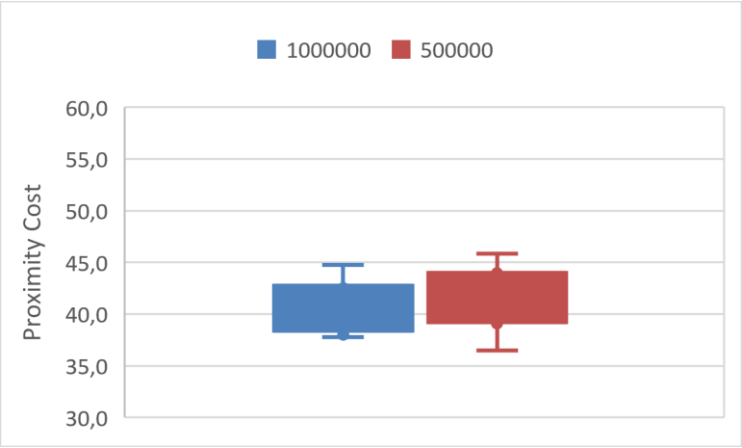
Perbandingan	UC 10	UC 11
Proximity Cost Minimum	37.795	36.503
Proximity Cost Maksimum	44.735	45.824
Rata-Rata Proximity Cost	40.824	41.686

Gambar 6.16 menunjukkan diagram batang dari perbandingan hasil uji coba 10 dan 11 yang dijalankan sebanyak 11 kali. Dari grafik tersebut, dapat dilihat bahwa hasil uji coba 9 dan 10 menunjukkan nilai *proximity cost* yang relatif sama. Pada beberapa iterasi, hasil uji coba 9 menunjukkan nilai *proximity cost* yang jauh lebih tinggi dibandingkan hasil uji coba 10. Akan tetapi, pada iterasi lain dapat ditemukan bahwa uji coba 10 juga dapat menunjukkan nilai *proximity cost* yang lebih tinggi dibandingkan uji coba 9.



Gambar 6.15. Perbandingan Hasil Uji Coba 9 dan 10

Sedangkan dari gambar 6.17, dapat dilihat bahwa uji coba 10 dan 11 mempunyai *range* nilai *proximity cost* yang hampir sama. Hal ini dikarenakan nilai *proximity cost* yang didapat juga tidak terpaut jauh. Nilai *proximity cost* maksimum yang didapat dari grafik tersebut adalah 45,824 yang merupakan hasil uji coba 11. Sedangkan nilai *proximity cost* minimum yang didapat dari grafik tersebut adalah 36,503 yang merupakan hasil dari uji coba 11. Dari grafik ini dapat disimpulkan bahwa jumlah iterasi yang digunakan menjadi tidak terlalu berpengaruh dengan jumlah antrian 10.



Gambar 6.16. *Box Plot* Perbandingan Hasil Uji Coba 9 dan 10

6.6.8. Perbandingan Hasil Uji Coba Dataset UTS dan UAS

Hasil yang akan dibandingkan adalah uji coba pada *dataset* UTS dan UAS. Perbandingan ini menggunakan jumlah antrian dan jumlah iterasi yang sama. Pada uji coba ini, jumlah antrian yang digunakan sebanyak 30 dan jumlah iterasi yang digunakan sebanyak 500.000. Parameter yang digunakan pada uji coba ini dipilih karena parameter tersebut dianggap memberikan hasil yang paling baik. Perbedaan yang terdapat pada *dataset* UTS dan UAS adalah jumlah mahasiswa, dimana pada *dataset* UAS jumlah mahasiswanya lebih sedikit dari *dataset* UTS. Tabel 6.29 menunjukkan perbedaan pada *dataset* yang digunakan.

Tabel 6.29. Perbandingan Uji Coba Dataset UTS dan UAS

Dataset	Mata Kuliah	Sesi	Mahasiswa
UTS	32	15	567
UAS	32	15	519

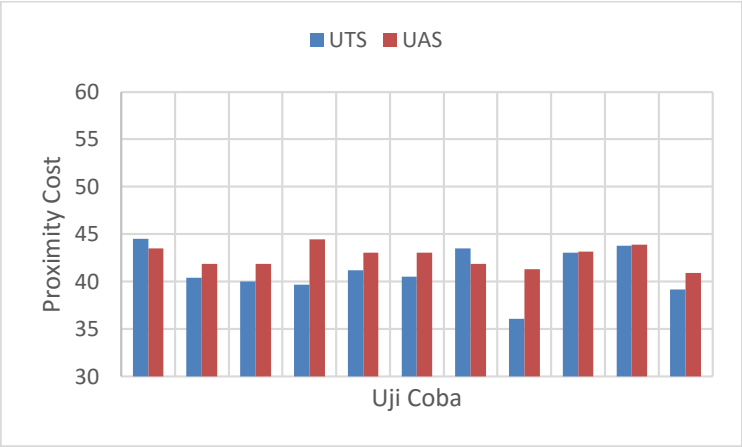
Kedua *dataset* tersebut dijalankan sebanyak 11 kali. Dari percobaan sebanyak 11 kali, didapatkan nilai *proximity cost* seperti yang ditampilkan pada tabel 6.32. Dapat dilihat pada tabel 6.30 bahwa *dataset* UTS memiliki nilai *proximity cost* yang lebih baik jika dibandingkan dengan hasil uji coba pada *dataset* UAS. Hal ini dikarenakan hasil uji coba pada *dataset*

UTS menunjukkan nilai *proximity cost* minimum, nilai *proximity cost* maksimum, dan nilai rata-rata *proximity cost* yang lebih rendah jika dibandingkan dengan nilai *proximity cost* dari hasil uji coba dengan *dataset* UAS.

Tabel 6.30. Perbandingan Hasil Uji Coba *Dataset* UTS dan UAS

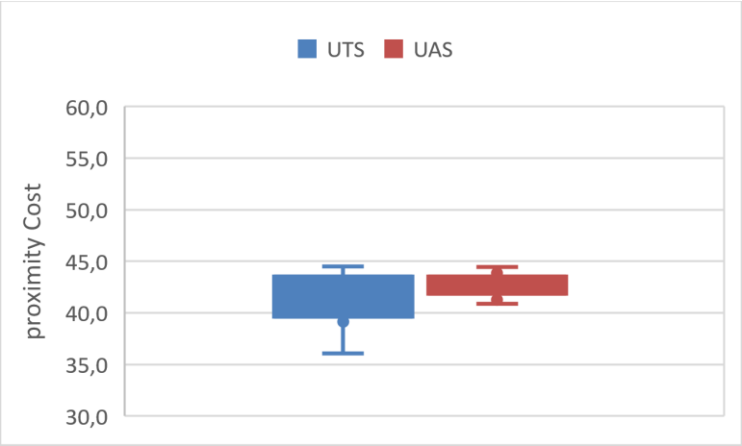
Perbandingan	UTS	UAS
Proximity Cost Minimum	36.083	40.896
Proximity Cost Maksimum	44.473	44.468
Rata-Rata Proximity Cost	41.062	42.618

Gambar 6.17 menunjukkan *diagram batang dari* perbandingan hasil uji coba dengan menggunakan *dataset* UTS dan UAS yang dijalankan sebanyak 11 kali.



Gambar 6.17. Perbandingan Hasil *Proximity Cost Dataset* UTS dan UAS

Sedangkan dari gambar 6.18, dapat dilihat bahwa *dataset* UTS mempunyai jangkauan nilai *proximity cost* yang lebih luas jika dibandingkan dengan nilai *proximity cost dataset* UAS. Namun, dengan menggunakan *dataset* UTS, nilai *proximity cost* yang didapat lebih rendah dibandingkan dengan menggunakan *dataset* UAS. Hal ini menunjukkan bahwa *dataset* UTS menghasilkan jadwal yang lebih baik.



Gambar 6.18. *Box Plot* Perbandingan Hasil Proximity Cost Dataset UTS dan UAS

6.7. Perbandingan Hasil Jadwal

Pada bagian ini akan dijelaskan mengenai perbandingan hasil dari jadwal yang diperoleh. Jadwal yang diperoleh terdiri dari tiga jenis, yaitu jadwal yang dibuat secara manual (jadwal manual), jadwal yang dihasilkan dari penerapan algoritma *greedy* (jadwal *greedy*), serta jadwal yang dihasilkan dari penerapan algoritma *late acceptance* (jadwal *late acceptance*). Jadwal *greedy* adalah yang didapatkan sebagai solusi inisial. Jadwal *greedy* tersebut adalah jadwal yang *feasible* dan sudah memenuhi batasan yang telah ditetapkan, yaitu tidak boleh ada jadwal yang bentrok dan tidak boleh melebihi kapasitas total ruang yang tersedia. Sedangkan jadwal *late acceptance* adalah optimasi dari jadwal *greedy*.

6.7.1. Perbandingan Hasil Jadwal UTS Manual dan Jadwal UTS *Greedy*

Perbandingan hasil jadwal UTS manual dan jadwal UTS *greedy* dapat dilihat pada tabel 6.33. Dari tabel 6.33 terdapat 3 mata kuliah yang mempunyai slot / sesi ujian yang sama, yaitu mata kuliah ke-1, ke-5, dan ke-6. Dari tabel tersebut juga terlihat

bahwa hampir seluruh mata kuliah dijadwalkan dengan slot / sesi ujian yang berbeda. Hal ini tentunya dilakukan untuk memenuhi batasan yang ditetapkan, yaitu tidak boleh ada jadwal yang bentrok dan tidak boleh melebihi kapasitas ruang yang tersedia.

Tabel 6.31. Jadwal UTS Manual dan Jadwal UTS *Greedy*

Kode Mata Kuliah	Jadwal UTS Manual	Jadwal UTS *Greedy*
0001	1	1
0002	15	2
0003	2	14
0004	7	11
0005	13	13
0006	5	5
0007	14	3
0008	1	6
0009	10	8
0010	4	1
0011	8	4
0012	4	3
0013	14	1
0014	11	7
0015	2	12
0016	7	11
0017	1	6
0018	6	5
0019	13	2
0020	3	8
0021	5	10
0022	12	9
0023	10	7
0024	9	11
0025	15	12
0026	2	13
0027	10	14
0028	9	7
0029	4	1
0030	8	4

Kode Mata Kuliah	Jadwal UTS Manual	Jadwal UTS *Greedy*
0031	11	15
0032	7	12

Jika dibandingkan dari nilai *proximity cost* yang diperoleh, maka jadwal UTS manual memiliki nilai *proximity cost* yang lebih tinggi jika dibandingkan dengan jadwal UTS *greedy*. Hal ini dikarenakan jadwal UTS manual mendapatkan penalti lebih karena banyak jadwal yang bentrok. Jadwal manual ini juga dinilai tidak *feasible*, karena melanggar batasan yang ditetapkan. Sedangkan pada jadwal UTS *greedy*, jadwal yang dihasilkan adalah jadwal yang sudah *feasible*, artinya jadwal tersebut sudah memenuhi batasan, dimana tidak ada jadwal yang bentrok dan tidak melebihi kapasitas total ruang yang tersedia. Dapat disimpulkan bahwa jadwal UTS *greedy* yang dihasilkan mampu menghasilkan nilai *proximity cost* yang hampir 2 kali lipat lebih rendah jika dibandingkan dengan jadwal UTS manual.

6.7.2. Perbandingan Hasil Jadwal UTS *Greedy* dan Jadwal UTS *Late Acceptance*

Perbandingan hasil jadwal UTS *greedy* dan jadwal UTS *late acceptance* dapat dilihat pada tabel 6.34. Dari tabel 6.34 terdapat 3 mata kuliah yang mempunyai slot / sesi ujian yang sama, yaitu mata kuliah ke-3, ke-14, dan ke-28. Dari tabel tersebut juga terlihat bahwa hampir seluruh mata kuliah dijadwalkan dengan slot / sesi ujian yang berbeda.

Tabel 6.32. Jadwal UTS *Greedy* dan Jadwal UTS *Late Acceptance*

Kode Mata Kuliah	Jadwal UTS *Greedy*	Jadwal UTS *Late Acceptance*
0001	1	7
0002	2	3
0003	14	14
0004	11	5
0005	13	11
0006	5	8

Kode Mata Kuliah	Jadwal UTS *Greedy*	Jadwal UTS *Late Acceptance*
0007	3	10
0008	6	2
0009	8	6
0010	1	3
0011	4	14
0012	3	12
0013	1	9
0014	7	7
0015	12	1
0016	11	5
0017	6	15
0018	5	2
0019	2	3
0020	8	6
0021	10	8
0022	9	13
0023	7	1
0024	11	12
0025	12	5
0026	13	4
0027	14	1
0028	7	7
0029	1	4
0030	4	11
0031	15	7
0032	12	5

Jika dibandingkan dari nilai *proximity cost* yang diperoleh, maka jadwal UTS *greedy* memiliki nilai *proximity cost* yang lebih tinggi jika dibandingkan dengan jadwal UTS *late acceptance.* Hal ini dikarenakan jadwal UTS *greedy* adalah jadwal yang sudah *feasible,* tapi belum optimal. Sedangkan pada jadwal UTS *late acceptance,* jadwal yang dihasilkan adalah jadwal yang sudah *feasible* dan sudah optimal.

6.7.3. Perbandingan Hasil Jadwal UTS Manual dan Jadwal UTS *Late Acceptance*

Perbandingan hasil jadwal UTS manual dan jadwal UTS *late acceptance* dapat dilihat pada tabel 6.35. Dari tabel 6.35 hanya terdapat 1 mata kuliah yang mempunyai slot / sesi ujian yang sama, yaitu mata kuliah ke-29. Dari tabel tersebut juga terlihat bahwa hampir seluruh mata kuliah dijadwalkan dengan slot / sesi ujian yang berbeda.

Tabel 6.33. Jadwal UTS Manual dan Jadwal UTS *Late Acceptance*

Kode Mata Kuliah	Jadwal UTS Manual	Jadwal UTS *Late Acceptance*
0001	1	7
0002	15	3
0003	2	14
0004	7	5
0005	13	11
0006	5	8
0007	14	10
0008	1	2
0009	10	6
0010	4	3
0011	8	14
0012	4	12
0013	14	9
0014	11	7
0015	2	1
0016	7	5
0017	1	15
0018	6	2
0019	13	3
0020	3	6
0021	5	8
0022	12	13
0023	10	1
0024	9	12
0025	15	5
0026	2	4
0027	10	1

Kode Mata Kuliah	Jadwal UTS Manual	Jadwal UTS *Late Acceptance*
0028	9	7
0029	4	4
0030	8	11
0031	11	7
0032	7	5

Jika dibandingkan dari nilai *proximity cost* yang diperoleh, maka jadwal UTS manual memiliki nilai *proximity cost* yang jauh lebih tinggi jika dibandingkan dengan jadwal UTS *late acceptance.*

Hal ini dikarenakan jadwal UTS manual mendapatkan penalti lebih karena banyak jadwal yang bentrok. Jadwal manual ini juga dinilai tidak *feasible*, karena melanggar batasan yang ditetapkan. Sedangkan pada jadwal UTS *late acceptance*, jadwal yang dihasilkan adalah jadwal yang sudah *feasible* dan sudah optimal, artinya jadwal tersebut sudah memenuhi batasan, dimana tidak ada jadwal yang bentrok dan tidak melebihi kapasitas total ruang yang tersedia.

6.7.4. Perbandingan Hasil Jadwal UAS Manual dan Jadwal UAS *Greedy*

Perbandingan hasil jadwal UAS manual dan jadwal UAS *greedy* dapat dilihat pada tabel 6.36. Dari tabel 6.36 terdapat 2 mata kuliah yang mempunyai slot / sesi ujian yang sama, yaitu mata kuliah ke-11 dan ke-32. Dari tabel tersebut juga terlihat bahwa hampir seluruh mata kuliah dijadwalkan dengan slot / sesi ujian yang berbeda.

Tabel 6.34. Jadwal UAS Manual dan Jadwal UAS *Greedy*

Kode Mata Kuliah	Jadwal UAS Manual	Jadwal UAS *Greedy*
0001	1	2
0002	8	11
0003	2	13
0004	7	1
0005	10	12

Kode Mata Kuliah	Jadwal UAS Manual	Jadwal UAS *Greedy*
0006	3	4
0007	11	2
0008	1	5
0009	11	6
0010	4	1
0011	8	8
0012	4	2
0013	11	3
0014	4	10
0015	2	9
0016	7	11
0017	1	4
0018	4	5
0019	10	1
0020	3	6
0021	5	9
0022	11	7
0023	8	10
0024	9	11
0025	4	3
0026	8	3
0027	2	10
0028	9	3
0029	2	12
0030	12	8
0031	12	13
0032	12	12

Jika dibandingkan dari nilai *proximity cost* yang diperoleh, maka jadwal UAS manual memiliki nilai *proximity cost* yang lebih tinggi jika dibandingkan dengan jadwal UAS *greedy*. Hal ini dikarenakan jadwal UAS manual mendapatkan penalti lebih karena banyak jadwal yang bentrok. Jadwal manual ini juga dinilai tidak *feasible*, karena melanggar batasan yang ditetapkan. Sedangkan pada jadwal UAS *greedy*, jadwal yang dihasilkan adalah jadwal yang sudah *feasible*, artinya jadwal tersebut sudah memenuhi batasan, dimana tidak ada jadwal

yang bentrok dan tidak melebihi kapasitas total ruang yang tersedia.

6.7.5. Perbandingan Hasil Jadwal UAS *Greedy* dan Jadwal UAS *Late Acceptance*

Perbandingan hasil jadwal UTS manual dan jadwal UTS *greedy* dapat dilihat pada tabel 6.37. Dari tabel 6.37 hanya terdapat 1 mata kuliah yang mempunyai slot / sesi ujian yang sama, yaitu mata kuliah ke-26. Dari tabel tersebut juga terlihat bahwa hampir seluruh mata kuliah dijadwalkan dengan slot / sesi ujian yang berbeda.

Tabel 6.35. Jadwal UAS *Greedy* dan Jadwal UAS *Late Acceptance*

Kode Mata Kuliah	Jadwal UAS *Greedy*	Jadwal UAS *Late Acceptance*
0001	2	8
0002	11	2
0003	13	6
0004	1	13
0005	12	10
0006	4	4
0007	2	14
0008	5	1
0009	6	12
0010	1	12
0011	8	3
0012	2	14
0013	3	6
0014	10	11
0015	9	8
0016	11	5
0017	4	1
0018	5	4
0019	1	13
0020	6	7
0021	9	12
0022	7	9
0023	10	11
0024	11	8

Kode Mata Kuliah	Jadwal UAS *Greedy*	Jadwal UAS *Late Acceptance*
0025	3	6
0026	3	3
0027	10	11
0028	3	7
0029	12	3
0030	8	2
0031	13	5
0032	12	10

Jika dibandingkan dari nilai *proximity cost* yang diperoleh, maka jadwal UAS *greedy* memiliki nilai *proximity cost* yang lebih tinggi jika dibandingkan dengan jadwal UAS *late acceptance.* Hal ini dikarenakan jadwal UAS *greedy* adalah jadwal yang sudah *feasible,* tapi belum optimal. Sedangkan pada jadwal UAS *late acceptance,* jadwal yang dihasilkan adalah jadwal yang sudah *feasible* dan sudah optimal.

6.7.6. Perbandingan Hasil Jadwal UAS Manual dan Jadwal UAS *Late Acceptance*

Perbandingan hasil jadwal UAS manual dan jadwal UAS *late acceptance* dapat dilihat pada tabel 6.38. Dari tabel 6.38 terdapat 2 mata kuliah yang mempunyai slot / sesi ujian yang sama, yaitu mata kuliah ke-5 dan ke-17. Dari tabel tersebut juga terlihat bahwa hampir seluruh mata kuliah dijadwalkan dengan slot / sesi ujian yang berbeda.

Tabel 6.36. Jadwal UAS Manual dan Jadwal UAS *Late Acceptance*

Kode Mata Kuliah	Jadwal UAS Manual	Jadwal UAS *Late Acceptance*
0001	1	8
0002	8	2
0003	2	6
0004	7	13
0005	10	10
0006	3	4
0007	11	14
0008	1	1

Kode Mata Kuliah	Jadwal UAS Manual	Jadwal UAS *Late Acceptance*
0009	11	12
0010	4	12
0011	8	3
0012	4	14
0013	11	6
0014	4	11
0015	2	8
0016	7	5
0017	1	1
0018	4	4
0019	10	13
0020	3	7
0021	5	12
0022	11	9
0023	8	11
0024	9	8
0025	4	6
0026	8	3
0027	2	11
0028	9	7
0029	2	3
0030	12	2
0031	12	5
0032	12	10

Jika dibandingkan dari nilai *proximity cost* yang diperoleh, maka jadwal UAS manual memiliki nilai *proximity cost* yang jauh lebih tinggi jika dibandingkan dengan jadwal UAS *late acceptance.*

Hal ini dikarenakan jadwal UAS manual mendapatkan penalti lebih karena banyak jadwal yang bentrok. Jadwal manual ini juga dinilai tidak *feasible,* karena melanggar batasan yang ditetapkan. Sedangkan pada jadwal UAS *late acceptance,* jadwal yang dihasilkan adalah jadwal yang sudah *feasible* dan sudah optimal, artinya jadwal tersebut sudah memenuhi batasan, dimana tidak ada jadwal yang bentrok dan tidak melebihi kapasitas total ruang yang tersedia.

6.8. Kesimpulan Hasil Uji Coba

Dari uji coba yang telah dilakukan dengan menggunakan beberapa skenario yang berbeda, dapat disimpulkan bahwa :

1. Pemilihan jumlah antrian pada algoritma *late acceptance* memiliki pengaruh terhadap nilai *proximity cost* dan jadwal ujian yang dihasilkan. Jumlah antrian yang lebih sedikit mampu menghasilkan jadwal ujian yang lebih optimal jika dibandingkan dengan jumlah antrian yang lebih banyak.
2. Pemilihan jumlah iterasi pada algoritma *late acceptance* juga memiliki pengaruh terhadap nilai *proximity cost* dan jadwal ujian yang dihasilkan. Jumlah iterasi yang lebih banyak memerlukan waktu yang lebih panjang untuk menghasilkan jadwal ujian, namun mampu menghasilkan jadwal ujian yang lebih optimal.
3. Dengan menggunakan *method swap* sebagai *low level heuristic*, nilai *proximity cost* yang dihasilkan menjadi semakin kecil (minimum). Sehingga jadwal ujian yang dihasilkan juga menjadi jadwal ujian yang lebih optimal.

BAB VII
KESIMPULAN DAN SARAN

Bab ini akan menjelaskan kesimpulan dari penelitian, beserta saran yang dapat bermanfaat untuk perbaikan di penelitian selanjutnya.

7.1. Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, maka dapat disimpulkan sebagai berikut :

1. Algoritma *greedy – late acceptance – hyper heuristic* dapat digunakan untuk membuat jadwal ujian di Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Jadwal yang dihasilkan adalah jadwal yang *feasible*, dimana jadwal tersebut sudah memenuhi batasan tidak ada jadwal yang bentrok dan tidak melebihi kapasitas ruang yang tersedia.
2. Algoritma *greedy – late acceptance – hyper heuristic* mampu menghasilkan jadwal ujian yang lebih optimal apabila dibandingkan dengan jadwal yang dihasilkan secara manual. Hal ini dibuktikan dengan nilai *proximity cost* yang lebih baik.
3. Dengan menggunakan algoritma *greedy – late acceptance – hyper heuristic*, pembuatan jadwal ujian menjadi semakin mudah dan cepat. Jika sebelumnya diperlukan waktu hampir dua minggu untuk membuat jadwal ujian, maka dengan adanya penjadwalan ujian otomatis ini, waktu yang diperlukan untuk membuat jadwal ujian hanya kurang dari 5 menit.
4. Pembuatan jadwal ujian otomatis dengan algoritma *greedy – late acceptance – hyper heuristic* dapat menghasilkan jadwal ujian yang *feasible* dan dapat mengoptimalkan jadwal ujian.

7.2. Saran

Saran yang diberikan berdasarkan hasil pengerjaan tugas akhir untuk penelitian selanjutnya yaitu :

1. Pada pengerjaan tugas akhir ini, setiap satu mata kuliah hanya dijadwalkan untuk satu slot dengan asumsi bahwa setiap mata kuliah melakukan UTS dan UAS. Pada saat pengerjaan diketahui terdapat mata kuliah yang mengganti UAS dengan demo atau presentasi *final project*, sehingga pada penelitian selanjutnya diharapkan mampu mengatasi hal ini.
2. Penelitian selanjutnya dapat juga ditambahkan mengenai penjadwalan ruang ujian.
3. Pada pemilihan *low level heuristic* bisa ditambahkan dengan metode lain seperti *warming-up approach* atau *step-by-step reduction* [17] untuk mengetahui pengaruh terhadap kinerja algoritma serta hasil akhir dari fungsi tujuan.

DAFTAR PUSTAKA

[1] M. W. Carter, G. Laporte dan S. Y. Lee, "Examination timetabling: Algorithmic strategies and applications," *Journal of the Operational Research Society,* vol. 47, no. 3, pp. 373-383, 1996.

[2] A. Muklason, "Solver Penjadwal Ujian Otomatis Dengan Algoritma Maximal Clique dan Hyper-heuristics," dalam *Seminar Nasional Teknologi Informasi, Komunikasi dan Industri (SNTIKI) 9, Fakultas Sains dan Teknologi, UIN Sultan Syarif Kasim Riau*, Pekanbaru, 2017.

[3] A. Muklason, A. J. Parkes, E. Özcan, B. McCollum dan P. McMullan, "Fairness in examination timetabling: Student preferences and extended formulations," *Elsevier,* no. 55, pp. 302-318, 2017.

[4] E. L. Lawler, Combinatorial Optimization: Networks and Matroids, Holt Rinehart and Winston, 1976.

[5] E. K. Burke, G. Kendall, M. Misir, E. Özcan dan E. Burke, "Applications to timetabling," dalam *Handbook of Graph Theory*, Citeseer, 2004, pp. 445-474.

[6] A. Schaerf, "A survey of automated timetabling," *Artificial Intelligence Review,* vol. 12, no. 2, pp. 87-127, April 1999.

[7] E. K. Burke, G. Kendall, M. Misir dan E. Özcan, "Monte Carlo hyper-heuristics for examination timetabling," *Annals of Operations Research,* vol. 196, no. 1, pp. 73-90, 2012.

[8] E. K. Burke, A. J. Eckersley, B. McCollum, S. Petrovic dan R. Qu, "Hybrid variable neighbourhood approaches to university exam timetabling," *European Journal of Operational Research,* vol. 206, no. 1, pp. 46-53, 2010.

[9] T. B. Cooper dan J. H. Kingston, "The complexity of timetable construction problems. In Edmund Burke and

Peter Ross, editors, Practice and Theory of Automated Timetabling," *Springer Berlin Heidelberg*, vol. 1153 of Lecture Notes in Computer Science, pp. 281-295, 1996.

[10] A. Muklason, "Hyper-heuristics and Fairness in Examination Timetabling Problems," The University of Nottingham, Nottingham, 2017.

[11] University of Toronto, "University of Toronto Benchmark Data," 1996. [Online]. Available: ftp://ftp.mie.utoronto.ca/pub/carter/testprob. [Diakses 26 September 2017].

[12] R. Munir, "Algoritma Greedy," Program Studi Informatika, Sekolah Teknik Elektro dan Informatika ITB, Bandung, 2004.

[13] R. Hartanto, "Algoritma Greedy dalam Strategi Permainan Centipede," Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika ITB, Bandung, 2017.

[14] E. Özcan, Y. Bykov, M. Birben dan E. K. Burke, "Examination Timetabling Using Late Acceptance Hyper-heuristics," dalam *IEEE Congress on Evolutionary Computation CEC '09*, 2009.

[15] E. K. Burke, G. Kendall, J. Newall, E. Hart, P. Ross dan S. Schulenburg, "Hyper-Heuristics: An emerging direction in modern search.In Fred Glover and GaryA. Kochenberger, editors, Handbook of Metaheuristics," *International Series in Operations Research & Management. Springer US.*, vol. 57, pp. 457-474, 2003.

[16] A. Arram, M. Ayob dan M. Zakree, "Comparative Study of Meta-Heuristic Approaches for Solving Traveling Salesman Problems," *Asian Journal of Applied Sciences*, vol. 7, no. 7, pp. 662-670, 2014.

[17] K. Chakhlevitch dan P. Cowling, "Choosing the Fittest Subset of Low Level Heuristics in a Hyperheuristiv Framework," University of Bradford, Bradford.

BIODATA PENULIS

Penulis mempunyai nama lengkap Putri Cahyaning Bwananesia, dilahirkan di Surabaya pada tanggal 08 Agustus 1992. Penulis menempuh pendidikan dasar di SDN Wonokusumo IX / 595 Surabaya. Setelah menamatkan sekolah dasar, penulis melanjutkan pendidikan tingkat menengah di SMP Negeri 11 Surabaya dan SMA Negeri 7 Surabaya. Pada tahun 2009, penulis diterima di Program Studi S1 Matematika, Fakultas Sains dan Teknologi, Universitas Airlangga Surabaya melalui jalur PMDK Prestasi. Penulis sempat menempuh pendidikan disana selama satu tahun, sebelum akhirnya mengikuti tes SNMPTN dan diterima di Jurusan Sistem Informasi, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya pada tahun 2010. Selama berkuliah di Departemen Sistem Informasi – FTIF – ITS, penulis aktif mengikuti kegiatan kepanitiaan, baik dalam lingkup jurusan, fakultas, maupun institut. Penulis juga pernah menjadi staf Departemen Hubungan Luar Himpunan Mahasiswa Sistem Informasi ITS dan staf Kementrian PSDM BEM ITS pada tahun 2011. Pada tahun 2012, penulis menjabat sebagai Bendahara I Himpunan Mahasiswa Sistem Informasi ITS. Selain kegiatan di dalam kampus, penulis juga berkegiatan di luar kampus dengan mendirikan komunitas *"Inspiring Youth Educators"*. Komunitas tersebut merupakan kumpulan pemuda yang peduli dan memiliki *passion* di bidang sosial pendidikan. Penulis pernah menerima "Beasiswa IKA ITS untuk Aktivis Mahasiswa" pada tahun 2013. Penulis dapat dihubungi melalui email : putribwananesia@gmail.com.

Halaman ini sengaja dikosongkan.

UCAPAN TERIMA KASIH

Penulis mengucapkan puji syukur ke hadirat Allah SWT yang telah memberikan kelancaran dan kemudahan bagi penulis untuk menyelesaikan Tugas Akhir ini. Penulis juga mengucapkan terima kasih kepada :

1. Kedua orang tua penulis, Ibu Joeli Hendri Koerniawianti dan Bapak Toto Sudjiwo yang selalu mendoakan dan mendukung penulis untuk menyelesaikan perkuliahan.
2. Adik penulis, Dinda Arum Mustikoweni, Rafli Akbar Januar Alif dan Luh Acintya Widyadana yang juga selalu mendoakan penulis untuk menyelesaikan perkuliahan.
3. Rekan satu tim penulis, Gigih Yudha Hutama dan Dian Kusumawardani yang telah menemani penulis dan bersedia diajak diskusi dalam pengerjaan tugas akhir.
4. Teman-teman FOXIS, khususnya Kartika Maulida Hindrayani, Febri Ari Wicaksono, Lutfi Cahyo Bintoro, Ahmad Faizun, dan Fasha Heikal yang telah memberikan dukungan bagi penulis.
5. Mas Ricky Asrul Sani selaku admin laboratorium Rekayasa Data dan Inteligensi Bisnis (RDIB) yang telah memberikan kemudahan bagi penulis dan bersedia menjawab pertanyaan-pertanyaan seputar administrasi lab.
6. Teman-teman dan adik-adik dari laboratorium Rekayasa Data dan Inteligensi Bisnis (RDIB) yang telah memberikan dukungan dan bantuan kepada penulis.

Serta kepada semua pihak-pihak yang tidak dapat penulis sebutkan satu per satu. Penulis berharap tugas akhir ini dapat memberikan manfaat bagi pihak yang membutuhkan.

Halaman ini sengaja dikosongkan.

LAMPIRAN A : *DATASET*

Dataset yang digunakan dalam tugas akhir ini terdiri dari *dataset* UTS dan UAS. Berikut lampiran *dataset* yang digunakan dalam format .stu.

A.1. Dataset UTS

Baris ke-	Data
1	0023
2	0010 0014
3	0032
4	0013 0022
5	0007 0011 0013 0014 0025 0026 0027
6	0031
7	0021 0022 0025
8	0013 0017 0018 0023 0024
9	0011
10	0031
11	0007 0011 0018 0020 0022 0024 0025
12	0007 0017 0018 0023
13	0023 0029
14	0007 0017 0019 0021 0022 0024 0029
15	0025
16	0007 0027
17	0031
18	0007 0011 0013 0014 0024
19	0009 0014 0015 0018 0019
20	0009 0011 0012 0017 0019 0022 0032
21	0006 0013 0030
22	0017

23	0022 0027
24	0013 0018 0019 0023 0032
25	0004 0011 0014 0017 0020
26	0013 0022 0031
27	0021 0022 0030
28	0013 0022 0024
29	0032
30	0017
31	0031
32	0025 0031
33	0017 0031
34	0027
35	0018 0021 0022 0030 0031
36	0032
37	0031
38	0013 0019 0020 0021 0032
39	0031
40	0017 0028
41	0032
42	0031
43	0031
44	0032
45	0027 0031
46	0007 0018 0021 0030 0031
47	0007 0017 0021 0022 0031 0032
48	0032
49	0017 0020 0021 0022
50	0032
51	0020
52	0031

53	0030 0031
54	0012
55	0011 0012 0016 0017 0021 0023
56	0025
57	0002 0004 0006 0007 0008 0014
58	0006 0009 0013
59	0032
60	0007 0017 0028
61	0030 0031
62	0024 0031
63	0023 0025 0030
64	0027 0031
65	0030 0031
66	0013 0017 0018 0019 0021 0030
67	0011 0013 0014 0019 0020 0024
68	0018 0027
69	0027
70	0030
71	0026 0027 0030 0031 0032
72	0017 0022 0023 0026 0032
73	0020 0022
74	0017 0021 0030 0031
75	0031 0032
76	0026 0030 0031 0032
77	0018 0031
78	0007 0017 0018 0021 0024 0032
79	0007 0017 0018 0021 0022 0028 0030
80	0007 0017 0021 0022 0031 0032
81	0031
82	0021

83	0025
84	0021
85	0017 0019 0021 0022 0023 0031
86	0030
87	0026
88	0007 0031 0032
89	0030 0031
90	0021 0031
91	0031
92	0017 0018 0022
93	0017
94	0007 0017 0018 0019 0020 0021 0022 0030
95	0007 0017 0018 0019 0020 0021 0022
96	0013 0018 0020 0021 0022 0024
97	0007 0017 0018 0019 0022 0023
98	0007 0017 0018 0019 0020 0021 0022
99	0007 0017 0018 0020 0021 0022 0030
100	0007 0017 0018 0019 0020 0021 0022
101	0007 0017 0018 0020 0021 0022 0024 0026
102	0007 0017 0018 0020 0021 0022 0026 0030
103	0007 0017 0018 0020 0021 0022 0025 0027
104	0007 0017 0018 0020 0021 0022 0026
105	0007 0017 0019 0020 0021 0022 0025 0029
106	0007 0017 0018 0019 0020 0021 0022 0024
107	0007 0017 0018 0019 0020 0021 0022
108	0007 0017 0018 0019 0020 0021 0022
109	0007 0017 0018 0019 0020 0021 0022 0027
110	0007 0017 0018 0019 0020 0021 0022
111	0007 0017 0018 0020 0021 0022 0027
112	0017 0018 0021 0022 0027 0030

113	0007 0017 0018 0019 0020 0021 0022
114	0007 0017 0018 0020 0021 0022 0030
115	0007 0017 0018 0020 0021 0022 0023 0025
116	0007 0013 0017 0018 0020 0021 0022
117	0007 0017 0018 0019 0020 0021 0022
118	0007 0017 0020 0021 0022 0030 0032
119	0017 0018 0019 0020 0021 0022 0027
120	0007 0017 0018 0020 0021 0022 0032
121	0007 0017 0018 0019 0020 0021 0022 0024
122	0007 0017 0018 0019 0020 0021 0022 0030
123	0007 0017 0018 0020 0021 0022 0032
124	0007 0017 0018 0019 0020 0021 0022
125	0007 0017 0018 0020 0021 0022 0023 0026
126	0007 0017 0018 0020 0021 0022 0024 0026
127	0007 0017 0018 0020 0021 0022 0023 0032
128	0007 0017 0018 0019 0020 0021 0022 0029
129	0007 0017 0020 0021 0022 0026
130	0013 0014 0017 0018 0020
131	0007 0017 0018 0019 0020 0021 0022
132	0007 0017 0018 0020 0021 0022 0025 0029
133	0008 0011 0012 0014 0016
134	0007 0017 0018 0019 0020 0021 0022 0024
135	0007 0017 0018 0019 0020 0021 0022 0023
136	0007 0017 0018 0020 0021 0022 0024
137	0011 0012 0013 0014 0020 0022
138	0007 0017 0018 0020 0021 0022 0032
139	0007 0017 0018 0020 0021 0022 0027 0030
140	0007 0017 0018 0020 0021 0022 0026 0027
141	0007 0017 0018 0020 0021 0022 0023
142	0007 0017 0018 0020 0021 0022 0024 0029

143	0007 0017 0018 0019 0020 0021 0022 0026
144	0007 0017 0018 0019 0020 0021 0022
145	0007 0017 0018 0020 0021 0022 0026 0027
146	0007 0017 0018 0020 0021 0022 0030
147	0007 0017 0018 0019 0020 0021 0022 0023
148	0007 0017 0018 0020 0021 0022 0023 0032
149	0007 0017 0018 0020 0021 0022 0027 0030
150	0007 0017 0018 0020 0021 0022 0027 0030
151	0007 0017 0018 0020 0021 0022 0027 0030
152	0007 0017 0018 0020 0021 0022 0032
153	0007 0017 0018 0021 0022 0030 0032
154	0007 0017 0018 0020 0021 0022 0023 0029
155	0007 0017 0018 0020 0021 0022 0030
156	0007 0017 0018 0020 0021 0022 0025
157	0007 0017 0018 0020 0021 0022 0023 0032
158	0007 0017 0018 0020 0021 0022 0032
159	0007 0014 0016 0017 0020 0022
160	0007 0017 0018 0020 0021 0022 0025
161	0007 0017 0018 0020 0021 0022 0023 0031
162	0007 0017 0018 0020 0021 0022 0023 0029
163	0007 0017 0018 0019 0020 0021 0022 0023
164	0007 0017 0018 0019 0020 0021 0022 0029
165	0007 0017 0018 0020 0021 0022 0032
166	0007 0017 0018 0019 0020 0021 0022 0026
167	0007 0017 0018 0020 0021 0022 0032
168	0007 0017 0018 0020 0021 0022 0026 0027
169	0007 0017 0018 0020 0021 0022 0029
170	0007 0017 0018 0021 0022 0025 0030
171	0007 0017 0018 0020 0021 0022 0024 0027
172	0007 0017 0018 0020 0021 0022 0027 0030

173	0007 0017 0018 0019 0020 0021 0022
174	0007 0017 0018 0019 0020 0021 0022
175	0011 0013 0014 0017 0020 0024
176	0007 0020 0021
177	0017 0018 0020 0021 0022 0029
178	0017 0018 0019 0020 0021 0022
179	0017 0018 0019 0020 0021 0022
180	0017 0018 0019 0020 0021 0022
181	0013 0017 0019 0020 0021 0022
182	0007 0017 0018 0019 0021 0022 0026
183	0017 0018 0020 0021 0022 0025 0029
184	0017 0019 0020 0021 0022 0030 0031
185	0007 0014 0017 0018 0019 0020 0022
186	0007 0017 0018 0020 0021 0022 0032
187	0007 0017 0018 0020 0021 0022 0023 0024
188	0007 0017 0018 0020 0021 0022 0023 0030
189	0007 0017 0020 0021 0022 0026 0027
190	0007 0017 0018 0020 0021 0022 0027
191	0007 0017 0018 0020 0021 0022 0024
192	0007 0017 0018 0020 0021 0022 0026
193	0017 0018 0020 0021 0022 0023
194	0011 0012 0014 0015 0019 0020
195	0017 0018 0019 0021 0022 0028 0029
196	0007 0017 0018 0020 0021 0022 0024 0030
197	0007 0017 0018 0019 0020 0021 0022
198	0012 0014 0017 0018 0019 0020 0022
199	0007 0017 0018 0020 0021 0022 0027
200	0007 0017 0018 0020 0021 0022 0030 0032
201	0007 0017 0018 0019 0020 0021 0022
202	0007 0017 0020 0021 0022 0023 0025 0029

203	0007 0017 0018 0020 0021 0022 0024 0027
204	0007 0017 0018 0020 0021 0022 0027 0030
205	0007 0017 0018 0019 0020 0021 0022
206	0007 0017 0018 0019 0020 0021 0022 0029
207	0007 0017 0018 0020 0021 0022 0026 0027
208	0007 0017 0018 0019 0020 0021 0022
209	0007 0017 0018 0020 0021 0022 0024 0029
210	0007 0017 0018 0020 0021 0022 0026 0027
211	0007 0017 0018 0020 0021 0022 0026 0027
212	0011 0013 0019 0020 0021 0022
213	0007 0017 0018 0020 0021 0022 0025 0027
214	0007 0017 0018 0019 0020 0021 0022
215	0007 0017 0018 0019 0020 0021 0022 0024
216	0007 0017 0018 0019 0020 0021 0022 0027
217	0011 0017 0018 0019 0020 0021 0022
218	0007 0017 0018 0020 0021 0022 0025
219	0007 0017 0018 0019 0020 0021 0022
220	0007 0017 0018 0019 0020 0021 0022
221	0007 0017 0018 0019 0020 0021 0022
222	0007 0017 0018 0019 0020 0021 0022
223	0007 0017 0018 0019 0020 0021 0022
224	0007 0017 0018 0020 0021 0022 0026 0027
225	0007 0017 0018 0020 0021 0022 0032
226	0017 0018 0019 0020 0021 0022 0025
227	0007 0017 0018 0020 0021 0022 0025 0030
228	0007 0017 0018 0020 0021 0022 0027
229	0017 0018 0019 0020 0021 0022 0030
230	0007 0017 0018 0019 0020 0021 0022
231	0007 0017 0018 0019 0020 0021 0022
232	0011 0013 0014 0018 0020 0022

233	0007 0017 0018 0021 0022 0030 0031
234	0007 0017 0018 0019 0020 0021 0022
235	0011 0014 0018 0020 0022
236	0007 0017 0018 0020 0021 0022 0030
237	0007 0017 0018 0020 0021 0022 0026 0030
238	0007 0016 0017 0018 0019 0021 0022
239	0007 0017 0018 0020 0021 0022 0027
240	0007 0017 0018 0019 0020 0021 0022
241	0007 0017 0018 0019 0021 0022 0032
242	0007 0017 0018 0021 0022 0024 0025 0027
243	0007 0017 0018 0021 0022 0029 0030
244	0017 0018 0019 0021 0022 0024 0029
245	0007 0017 0018 0020 0021 0022 0032
246	0007 0017 0018 0021 0022 0026 0030
247	0007 0017 0018 0021 0022 0026 0032
248	0007 0017 0018 0020 0021 0022 0030
249	0007 0017 0018 0021 0022 0024 0025 0027
250	0007 0017 0018 0020 0021 0022 0024 0025
251	0007 0017 0019 0020 0021 0022 0030
252	0007 0017 0019 0020 0021 0022 0030
253	0007 0017 0018 0019 0020 0021 0022
254	0017 0018 0019 0020 0021 0022 0031
255	0007 0017 0018 0020 0021 0022 0027
256	0017 0018 0020 0021 0022 0027 0030
257	0011 0018 0019 0020 0022
258	0007 0017 0018 0019 0020 0021 0022
259	0017 0018 0019 0020 0021 0022 0032
260	0017 0018 0020 0021 0022 0030 0031
261	0017 0018 0019 0020 0021 0022 0032
262	0012 0017 0018 0019 0020 0021 0022

263	0017 0018 0019 0020 0021 0022 0026
264	0007 0017 0018 0019 0020 0021 0022 0025
265	0007 0017 0018 0021 0022 0026 0030
266	0007 0017 0018 0020 0021 0022 0023 0032
267	0007 0017 0018 0019 0020 0021 0022 0025
268	0011 0013 0019 0020 0021 0022
269	0007 0017 0018 0019 0020 0021 0022 0023
270	0008 0011 0013 0015 0019
271	0007 0017 0018 0019 0020 0021 0022 0024
272	0017 0018 0019 0020 0021 0022 0024 0025
273	0011 0012 0013 0014 0015 0019
274	0011 0012 0013 0014 0015 0016
275	0004 0011 0012 0013 0014 0015
276	0011 0012 0013 0014 0015
277	0011 0012 0013 0014 0015 0019
278	0011 0012 0013 0014 0015 0019
279	0011 0012 0013 0014 0015 0019
280	0011 0012 0013 0014 0015 0019
281	0011 0012 0013 0014 0015
282	0011 0012 0013 0014 0015 0019
283	0008 0011 0012 0013 0014 0015
284	0011 0012 0013 0014 0015
285	0011 0012 0013 0014 0015 0019
286	0011 0012 0013 0014 0015 0019
287	0011 0012 0013 0014 0015
288	0011 0012 0013 0014 0015 0019
289	0011 0012 0013 0014 0015
290	0002 0012 0014 0016 0018
291	0011 0012 0013 0014 0015 0019
292	0011 0012 0013 0014 0015

293	0011 0012 0013 0014 0015
294	0011 0012 0013 0014 0015
295	0011 0012 0013 0014 0015
296	0011 0012 0013 0014 0015 0019
297	0011 0012 0013 0014 0015 0016
298	0011 0012 0013 0014 0015 0016
299	0011 0012 0013 0014 0015
300	0011 0012 0013 0014 0015
301	0011 0012 0013 0014 0015 0016
302	0011 0012 0013 0014 0015 0019
303	0011 0012 0013 0014 0015
304	0011 0012 0013 0014 0015 0019
305	0011 0012 0013 0014 0015
306	0011 0012 0013 0014 0015
307	0011 0012 0013 0014 0015
308	0011 0012 0013 0014 0015
309	0011 0012 0013 0014 0015 0019
310	0011 0012 0013 0014 0015
311	0011 0012 0013 0014 0015 0016
312	0011 0012 0013 0014 0015
313	0011 0012 0013 0014 0015
314	0011 0012 0013 0014 0015
315	0011 0012 0013 0014 0015
316	0011 0012 0013 0014 0015 0019
317	0011 0012 0013 0014 0015
318	0011 0012 0013 0014 0015
319	0011 0012 0013 0014 0015 0020
320	0011 0012 0013 0014 0015 0019
321	0011 0012 0013 0014 0015
322	0011 0012 0013 0014 0015 0019

323	0011 0012 0013 0014 0015
324	0011 0012 0013 0014 0015 0016
325	0011 0012 0013 0014 0015 0019
326	0011 0012 0013 0014 0015
327	0011 0012 0013 0014 0015 0019
328	0011 0012 0013 0014 0015
329	0008 0011 0013 0014 0015
330	0011 0012 0013 0014 0015
331	0011 0012 0013 0014 0015
332	0011 0012 0013 0014 0015
333	0011 0012 0013 0014 0015
334	0011 0012 0013 0014 0015
335	0011 0012 0013 0014 0015
336	0011 0012 0013 0014 0015 0019
337	0011 0012 0013 0014 0015
338	0011 0012 0013 0014 0015 0019
339	0011 0012 0013 0014 0015
340	0011 0012 0013 0014 0015 0019
341	0011 0012 0013 0014 0015
342	0011 0012 0013 0014 0015
343	0011 0012 0013 0014 0015 0019
344	0011 0012 0013 0014 0015 0019
345	0011 0012 0013 0014 0015 0019
346	0011 0012 0013 0014 0015 0016
347	0011 0012 0013 0014 0015
348	0011 0012 0013 0014 0015
349	0011 0012 0013 0014 0015
350	0011 0012 0013 0014 0015
351	0011 0012 0013 0014 0015 0019
352	0011 0012 0013 0014 0015 0019

353	0011 0012 0013 0014 0015 0019
354	0011 0012 0013 0014 0015 0019
355	0011 0012 0013 0014 0015 0016
356	0011 0012 0013 0014 0015 0016
357	0011 0012 0013 0014 0015
358	0011 0012 0013 0014 0015
359	0011 0012 0013 0014 0015 0019
360	0011 0012 0013 0014 0015
361	0011 0012 0013 0014 0015 0019
362	0011 0012 0013 0014 0015
363	0011 0012 0013 0014 0015 0019
364	0006 0008 0011 0012 0015 0019
365	0011 0012 0013 0014 0015
366	0011 0012 0013 0014 0015
367	0008 0011 0012 0013 0014 0015
368	0011 0012 0013 0014 0015 0019
369	0006 0011 0012 0014 0015
370	0011 0012 0013 0014 0015
371	0011 0012 0013 0014 0015
372	0011 0012 0013 0014 0015
373	0011 0012 0013 0014 0015 0019
374	0011 0012 0013 0014 0015 0016
375	0011 0012 0013 0014 0015 0019
376	0011 0012 0013 0014 0015 0019
377	0008 0011 0012 0013 0015
378	0011 0012 0013 0014 0015
379	0011 0012 0013 0014 0015 0019
380	0011 0012 0013 0014 0015 0016
381	0011 0012 0013 0014 0015
382	0002 0006 0012 0014

383	0011 0012 0013 0014 0015
384	0011 0012 0013 0014 0015
385	0002 0004 0005 0014
386	0006 0008 0009 0012 0016
387	0011 0012 0013 0014 0015
388	0011 0012 0013 0014 0015 0019
389	0011 0012 0013 0014 0015 0019 0020
390	0011 0012 0013 0014 0015 0020
391	0011 0012 0013 0014 0015 0019
392	0011 0012 0013 0014 0015 0019
393	0011 0012 0013 0014 0015 0020
394	0002 0011 0012 0013 0014 0015
395	0011 0012 0013 0014 0015
396	0011 0012 0013 0014 0015
397	0011 0012 0013 0014 0015 0016
398	0011 0012 0013 0014 0015 0019
399	0011 0012 0013 0014 0015 0019
400	0011 0012 0013 0014 0015
401	0011 0012 0013 0014 0015
402	0011 0012 0013 0014 0015 0019
403	0011 0012 0013 0014 0015 0016
404	0011 0012 0013 0014 0015 0019
405	0011 0012 0013 0014 0015
406	0005 0011 0013 0014 0015
407	0011 0012 0013 0014 0015 0016
408	0011 0012 0013 0014 0015 0019
409	0011 0012 0013 0014 0015
410	0011 0012 0013 0014 0015 0018
411	0011 0012 0013 0014 0015 0019
412	0011 0012 0013 0014 0015 0016

413	0011 0012 0013 0014 0015 0019
414	0011 0012 0013 0014 0015 0019
415	0011 0012 0013 0014 0015
416	0011 0012 0013 0014 0015
417	0008 0011 0012 0013 0014 0015
418	0002 0003 0004 0005 0006
419	0002 0003 0004 0005
420	0002 0003 0004 0005 0006
421	0002 0003 0004 0005
422	0002 0003 0004 0005 0009
423	0002 0003 0004 0005 0008
424	0002 0003 0004 0005 0006
425	0002 0003 0004 0005 0006
426	0002 0003 0004 0005 0006
427	0002 0003 0004 0005 0006
428	0002 0003 0004 0005 0006
429	0002 0003 0004 0005 0008
430	0002 0003 0004 0005 0006
431	0002 0003 0004 0005 0006
432	0002 0003 0004 0005 0006
433	0002 0003 0004 0005 0009
434	0002 0003 0004 0005 0009
435	0002 0003 0004 0005
436	0002 0003 0004 0005
437	0002 0003 0004 0005 0006
438	0002 0003 0004 0005 0006
439	0002 0003 0004 0005 0006
440	0002 0003 0004 0005 0006
441	0002 0003 0004 0005 0006
442	0002 0003 0004 0005 0006

443	0002 0003 0004 0005
444	0002 0003 0004 0005 0006
445	0002 0003 0004 0005 0009
446	0002 0003 0004 0005 0009
447	0002 0003 0004 0005 0009
448	0002 0003 0004 0005 0006
449	0002 0003 0004 0005 0008
450	0002 0003 0004 0005 0006
451	0002 0003 0004 0005 0006
452	0002 0003 0004 0005 0006
453	0002 0003 0004 0005 0009
454	0002 0003 0004 0005
455	0002 0003 0004 0005
456	0002 0003 0004 0005 0006
457	0002 0003 0004 0005 0006
458	0002 0003 0004 0005
459	0002 0003 0004 0005
460	0002 0003 0004 0005 0009
461	0002 0003 0004 0005
462	0002 0003 0004 0005 0009
463	0002 0003 0004 0005 0006
464	0002 0003 0004 0005
465	0002 0003 0004 0005 0006
466	0002 0003 0004 0005 0006
467	0002 0003 0004 0005 0006
468	0002 0003 0004 0005
469	0002 0003 0004 0005
470	0002 0003 0004 0005 0006
471	0002 0003 0004 0005
472	0002 0003 0004 0005 0008

473	0002 0003 0004 0005
474	0002 0003 0004 0005
475	0002 0003 0004 0005
476	0002 0003 0004 0005 0006
477	0002 0003 0004 0005 0006
478	0002 0003 0004 0005
479	0002 0003 0004 0005 0009
480	0002 0003 0004 0005
481	0002 0003 0004 0005
482	0002 0003 0004 0005
483	0002 0003 0004 0005
484	0002 0003 0004 0005 0006
485	0002 0003 0004 0005 0009
486	0001 0002 0003 0004 0005
487	0001 0002 0003 0004 0005
488	0002 0003 0004 0005 0006
489	0002 0003 0004 0005 0009
490	0002 0003 0004 0005
491	0001 0002 0003 0004 0005
492	0002 0003 0004 0005
493	0002 0003 0004 0005
494	0002 0003 0004 0005 0009
495	0002 0003 0004 0005 0006
496	0002 0003 0004 0005 0006
497	0002 0003 0004 0005
498	0002 0003 0004 0005
499	0002 0003 0004 0005
500	0002 0003 0004 0005
501	0002 0003 0004 0005 0006
502	0002 0003 0004 0005 0006

503	0002 0003 0004 0005 0009
504	0002 0003 0004 0005 0006
505	0002 0003 0004 0005 0006
506	0002 0003 0004 0005 0006
507	0002 0003 0004 0005 0008
508	0002 0003 0004 0005 0009
509	0002 0003 0004 0005 0009
510	0002 0003 0004 0005 0009
511	0002 0003 0004 0005
512	0002 0003 0004 0005
513	0002 0003 0004 0005 0006
514	0002 0003 0004 0005 0006
515	0002 0003 0004 0005
516	0002 0003 0004 0005 0006
517	0002 0003 0004 0005 0006
518	0002 0003 0004 0005 0009
519	0002 0003 0004 0005 0006
520	0002 0003 0004 0005 0006
521	0002 0003 0004 0005
522	0002 0003 0004 0005
523	0002 0003 0004 0005 0009
524	0002 0003 0004 0005 0009
525	0002 0003 0004 0005
526	0002 0003 0004 0005 0008
527	0002 0003 0004 0005 0006
528	0002 0003 0004 0005 0006
529	0002 0003 0004 0005
530	0002 0003 0004 0005
531	0002 0003 0004 0005 0006
532	0002 0003 0004 0005

533	0002 0003 0004 0005
534	0002 0003 0004 0005
535	0002 0003 0004 0005 0006
536	0002 0003 0004 0005 0006
537	0002 0003 0004 0005
538	0002 0003 0004 0005 0006
539	0002 0003 0004 0005
540	0002 0003 0004 0005
541	0002 0003 0004 0005
542	0002 0003 0004 0005
543	0002 0003 0004 0005
544	0002 0003 0004 0005
545	0002 0003 0004 0005 0009
546	0002 0003 0004 0005 0006
547	0002 0003 0004 0005
548	0002 0003 0004 0005 0008
549	0002 0003 0004 0005
550	0002 0003 0004 0005
551	0002 0003 0004 0005 0009
552	0002 0003 0004 0005
553	0002 0003 0004 0005 0009
554	0002 0003 0004 0005
555	0002 0003 0004 0005 0006
556	0002 0003 0004 0005
557	0002 0003 0004 0005
558	0001 0002 0003 0004 0005
559	0002 0003 0004 0005 0009
560	0002 0003 0004 0005
561	0002 0003 0004 0005
562	0002 0003 0004 0005 0006

563	0002 0003 0004 0005 0006
564	0002 0003 0004 0005
565	0002 0003 0004 0005 0006
566	0002 0003 0004 0005 0006
567	0002 0003 0004 0005

A.2. Dataset UAS

Baris ke-	Data
1	0032
2	0013 0022
3	0021 0022 0025
4	0013 0017 0018 0023 0024
5	0011
6	0007 0017 0018 0023
7	0023 0029
8	0007 0017 0019 0021 0022 0024 0029
9	0025
10	0007 0027
11	0007 0011 0013 0014 0024
12	0009 0014 0015 0018 0019
13	0009 0011 0012 0017 0019 0022 0032
14	0006 0013
15	0017
16	0022 0027
17	0018 0019 0023 0032
18	0004 0011 0014 0017 0020
19	0013 0022 0031
20	0021 0022 0030
21	0013 0022 0024
22	0032

23	0027
24	0018 0021 0022 0030 0031
25	0013 0019 0020 0021 0032
26	0032
27	0027 0031
28	0007 0018 0021 0030 0031
29	0007 0017 0021 0022 0031 0032
30	0011 0012 0016 0023
31	0002 0004 0006 0007 0008 0014
32	0006 0009 0013
33	0007 0017 0028
34	0030 0031
35	0024 0031
36	0023 0025 0030
37	0013 0017 0018 0019 0030
38	0011 0013 0014 0019 0020 0024
39	0018 0027
40	0026 0027 0031 0032
41	0017 0022 0023 0026 0032
42	0020 0022
43	0031 0032
44	0026 0030 0031
45	0007 0017 0018 0021 0024 0032
46	0007 0017 0018 0021 0022 0028 0030
47	0007 0017 0022 0031 0032
48	0017 0019 0021 0022 0023 0031
49	0007 0031 0032
50	0030 0031
51	0031
52	0007 0017 0018 0019 0020 0021 0022 0030

53	0007 0017 0018 0019 0020 0021 0022
54	0013 0018 0020 0021 0022 0024
55	0007 0017 0018 0019 0021 0022 0023
56	0007 0017 0018 0019 0020 0021 0022
57	0007 0017 0018 0020 0021 0022 0030
58	0007 0017 0018 0019 0020 0021 0022
59	0007 0017 0018 0020 0021 0022 0024 0026
60	0007 0017 0018 0020 0021 0022 0026 0030
61	0007 0017 0018 0020 0021 0022 0025 0027
62	0007 0017 0018 0020 0021 0022 0026
63	0007 0017 0019 0020 0021 0022 0025 0029
64	0007 0017 0018 0019 0020 0021 0022 0024
65	0007 0017 0018 0019 0020 0021 0022
66	0007 0017 0018 0019 0020 0022
67	0007 0017 0018 0019 0020 0021 0022 0027
68	0007 0017 0018 0019 0020 0021 0022
69	0007 0017 0018 0020 0021 0022 0027
70	0017 0018 0021 0022 0027 0030
71	0007 0017 0018 0019 0020 0021 0022
72	0007 0017 0018 0020 0021 0022 0030
73	0007 0017 0018 0020 0021 0022 0023 0025
74	0007 0013 0017 0018 0020 0021 0022
75	0007 0017 0018 0019 0020 0021 0022
76	0007 0017 0020 0021 0022 0030 0032
77	0017 0018 0019 0020 0021 0022 0027
78	0007 0017 0018 0020 0021 0022 0032
79	0007 0017 0018 0019 0020 0021 0022 0024
80	0007 0017 0018 0019 0020 0021 0022 0030
81	0007 0017 0018 0020 0021 0022 0032
82	0007 0017 0018 0019 0020 0021 0022

83	0007 0017 0018 0020 0021 0022 0023 0026
84	0007 0017 0018 0020 0021 0022 0024 0026
85	0007 0017 0018 0020 0022 0023 0032
86	0007 0017 0018 0019 0020 0021 0022 0029
87	0007 0017 0020 0021 0022 0026
88	0013 0014 0017 0018 0020
89	0007 0017 0018 0019 0020 0021 0022
90	0007 0017 0018 0020 0021 0022 0025 0029
91	0008 0011 0012 0014 0016
92	0007 0017 0018 0019 0020 0021 0022 0024
93	0007 0017 0018 0019 0020 0021 0022 0023
94	0007 0017 0018 0020 0021 0022 0024
95	0011 0012 0013 0014 0020 0022
96	0017 0018 0020 0021 0022 0032
97	0007 0017 0018 0020 0021 0022 0027 0030
98	0007 0017 0018 0020 0021 0022 0026 0027
99	0007 0017 0018 0020 0021 0022 0023
100	0007 0017 0018 0020 0021 0022 0024 0029
101	0007 0017 0018 0019 0020 0021 0022 0026
102	0007 0017 0018 0019 0020 0021 0022
103	0007 0017 0018 0020 0021 0022 0026 0027
104	0007 0017 0018 0020 0021 0022 0030
105	0007 0017 0018 0019 0020 0021 0022 0023
106	0007 0017 0018 0020 0021 0022 0023 0032
107	0007 0017 0018 0020 0021 0022 0027 0030
108	0007 0017 0018 0020 0021 0022 0027 0030
109	0007 0017 0018 0020 0021 0022 0027 0030
110	0007 0017 0018 0020 0021 0022 0032
111	0007 0017 0018 0021 0022 0030 0032
112	0007 0017 0018 0020 0021 0022 0023 0029

113	0007 0017 0018 0020 0021 0022 0030
114	0007 0017 0018 0020 0022 0025
115	0007 0017 0018 0020 0021 0022 0023 0032
116	0007 0017 0018 0020 0021 0022 0032
117	0007 0014 0016 0017 0020 0022
118	0007 0017 0018 0020 0021 0022 0025
119	0007 0017 0018 0020 0021 0022 0023 0031
120	0007 0017 0018 0020 0021 0022 0023 0029
121	0007 0017 0018 0019 0020 0021 0022 0023
122	0007 0017 0018 0019 0020 0021 0022 0029
123	0007 0017 0018 0020 0021 0022 0032
124	0007 0017 0018 0019 0020 0021 0022 0026
125	0007 0017 0018 0020 0021 0022 0032
126	0007 0017 0018 0020 0021 0022 0026 0027
127	0007 0017 0018 0020 0021 0022 0029
128	0007 0017 0018 0021 0022 0025 0030
129	0007 0017 0018 0020 0021 0022 0024 0027
130	0007 0017 0018 0020 0021 0022 0027 0030
131	0007 0017 0018 0019 0020 0021 0022
132	0007 0017 0018 0019 0020 0022
133	0011 0013 0014 0017 0020 0024
134	0007 0020 0021
135	0017 0018 0020 0021 0022 0029
136	0017 0018 0019 0020 0021 0022
137	0017 0018 0019 0020 0021 0022
138	0017 0018 0019 0020 0022
139	0013 0017 0019 0020 0021 0022
140	0007 0017 0018 0019 0021 0022 0026
141	0017 0018 0020 0021 0022 0025 0029
142	0017 0019 0020 0021 0022 0030 0031

143	0007 0014 0017 0018 0019 0020 0022
144	0007 0017 0018 0020 0021 0022 0032
145	0007 0017 0018 0020 0021 0022 0023 0024
146	0007 0017 0018 0020 0021 0022 0023 0030
147	0007 0017 0020 0021 0022 0026 0027
148	0007 0017 0018 0020 0021 0022 0027
149	0007 0017 0018 0020 0021 0022 0024
150	0007 0017 0018 0020 0022 0026
151	0017 0018 0020 0021 0022 0023
152	0011 0012 0014 0015 0019 0020
153	0017 0018 0019 0021 0022 0028 0029
154	0007 0017 0018 0020 0021 0022 0024 0030
155	0007 0017 0018 0019 0020 0022
156	0012 0014 0017 0018 0019 0020 0022
157	0007 0017 0018 0020 0021 0022 0027
158	0007 0017 0018 0020 0021 0022 0030 0032
159	0007 0017 0018 0019 0020 0021 0022
160	0007 0017 0020 0021 0022 0023 0025 0029
161	0017 0018 0020 0021 0022 0024 0027
162	0007 0017 0018 0020 0021 0022 0027 0030
163	0007 0017 0018 0019 0020 0021 0022
164	0007 0017 0018 0019 0020 0021 0022 0029
165	0007 0017 0018 0020 0021 0022 0026 0027
166	0007 0017 0018 0019 0020 0021 0022
167	0007 0017 0018 0020 0021 0022 0024 0029
168	0007 0017 0018 0020 0022 0026 0027
169	0007 0017 0018 0020 0021 0022 0026 0027
170	0011 0019 0020 0021 0022
171	0007 0017 0018 0020 0021 0022 0025 0027
172	0007 0017 0018 0019 0020 0022

173	0007 0017 0018 0019 0020 0021 0022 0024
174	0007 0017 0018 0019 0020 0021 0022 0027
175	0011 0017 0018 0019 0020 0021 0022
176	0007 0017 0018 0020 0021 0022 0025
177	0007 0017 0018 0019 0020 0021 0022
178	0007 0017 0018 0019 0020 0021 0022
179	0007 0017 0018 0019 0020 0021 0022
180	0007 0017 0018 0019 0020 0021 0022
181	0007 0017 0018 0019 0020 0021 0022
182	0007 0017 0018 0020 0021 0022 0026 0027
183	0007 0017 0018 0020 0021 0022 0032
184	0017 0018 0019 0020 0021 0022 0025
185	0007 0017 0018 0020 0021 0022 0025 0030
186	0007 0017 0018 0020 0022 0027
187	0017 0018 0019 0020 0021 0022 0030
188	0007 0017 0018 0019 0020 0021 0022
189	0007 0017 0018 0019 0020 0022
190	0011 0013 0014 0018 0020 0022
191	0007 0017 0018 0021 0022 0030 0031
192	0007 0017 0018 0019 0020 0021 0022
193	0011 0014 0018 0020 0022
194	0007 0017 0018 0020 0021 0022 0030
195	0007 0017 0018 0020 0021 0022 0026 0030
196	0007 0016 0017 0018 0019 0021 0022
197	0007 0017 0018 0020 0021 0022 0027
198	0007 0017 0018 0019 0020 0022
199	0007 0017 0018 0019 0021 0022 0032
200	0007 0017 0018 0021 0022 0024 0025 0027
201	0007 0017 0018 0021 0022 0029 0030
202	0017 0018 0019 0021 0022 0024 0029

203	0007 0017 0018 0020 0021 0022 0032
204	0007 0017 0018 0021 0022 0026 0030
205	0007 0017 0018 0021 0022 0026 0032
206	0007 0017 0018 0020 0021 0022 0030
207	0007 0017 0018 0021 0022 0024 0025 0027
208	0007 0017 0018 0020 0021 0022 0024 0025
209	0007 0017 0019 0020 0021 0022 0030
210	0007 0017 0019 0020 0021 0022 0030
211	0007 0017 0018 0019 0020 0021 0022
212	0017 0018 0019 0020 0021 0022 0031
213	0007 0017 0018 0020 0021 0022 0027
214	0017 0018 0020 0021 0022 0027 0030
215	0011 0018 0019 0020 0022
216	0007 0017 0018 0019 0020 0021 0022
217	0017 0018 0019 0020 0021 0022 0032
218	0017 0018 0020 0021 0022 0030 0031
219	0017 0018 0019 0020 0021 0022 0032
220	0012 0017 0018 0019 0020 0021 0022
221	0017 0018 0019 0020 0021 0022 0026
222	0007 0017 0018 0019 0020 0021 0022 0025
223	0007 0017 0018 0021 0022 0026 0030
224	0007 0017 0018 0020 0021 0022 0023 0032
225	0007 0017 0018 0019 0020 0022 0025
226	0011 0013 0019 0020 0021 0022
227	0007 0017 0018 0019 0020 0022 0023
228	0008 0011 0013 0015 0019
229	0007 0017 0018 0019 0020 0021 0022 0024
230	0017 0018 0019 0020 0021 0022 0024 0025
231	0011 0012 0013 0014 0015 0019
232	0011 0012 0013 0014 0015 0016

233	0004 0011 0012 0013 0014 0015
234	0011 0012 0013 0014 0015
235	0011 0012 0013 0014 0015 0019
236	0011 0012 0013 0014 0015 0019
237	0011 0012 0013 0014 0015 0019
238	0011 0012 0013 0014 0015 0019
239	0011 0012 0013 0014 0015
240	0011 0012 0013 0014 0015 0019
241	0008 0011 0012 0013 0014 0015
242	0011 0012 0013 0014 0015
243	0011 0012 0013 0014 0015 0019
244	0011 0012 0013 0014 0015 0019
245	0011 0012 0013 0014 0015
246	0011 0012 0013 0014 0015 0019
247	0011 0012 0013 0014 0015
248	0011 0012 0013 0014 0015 0019
249	0011 0012 0014 0015
250	0011 0012 0014 0015
251	0011 0012 0014 0015
252	0011 0012 0014 0015
253	0011 0012 0014 0015 0019
254	0011 0012 0013 0014 0015 0016
255	0011 0012 0013 0014 0015 0016
256	0011 0012 0013 0014 0015
257	0011 0012 0013 0014 0015
258	0011 0012 0013 0014 0015 0016
259	0011 0012 0013 0014 0015 0019
260	0011 0012 0013 0014 0015 0019
261	0011 0012 0013 0014 0015
262	0011 0012 0013 0014 0015

263	0011 0012 0013 0014 0015
264	0011 0012 0013 0014 0015
265	0011 0012 0013 0014 0015 0019
266	0011 0012 0013 0014 0015
267	0011 0012 0013 0014 0015 0016
268	0011 0012 0013 0014 0015
269	0011 0012 0013 0014 0015
270	0011 0012 0013 0014 0015
271	0011 0012 0013 0014 0015
272	0011 0012 0013 0014 0015 0019
273	0011 0012 0014 0015
274	0011 0012 0014 0015
275	0011 0012 0014 0015 0020
276	0011 0012 0014 0015 0019
277	0011 0012 0014 0015
278	0011 0012 0013 0014 0015 0019
279	0011 0012 0013 0014 0015
280	0011 0012 0013 0014 0015 0016
281	0011 0012 0013 0014 0015 0019
282	0011 0012 0013 0014 0015
283	0011 0012 0013 0014 0015 0019
284	0011 0012 0013 0014 0015
285	0008 0011 0013 0014 0015
286	0011 0012 0013 0014 0015
287	0011 0012 0013 0014 0015
288	0011 0012 0013 0014 0015
289	0011 0012 0013 0014 0015
290	0011 0012 0013 0014 0015
291	0011 0012 0013 0014 0015
292	0011 0012 0013 0014 0015 0019

293	0011 0012 0013 0014 0015
294	0011 0012 0013 0014 0015 0019
295	0011 0012 0013 0014 0015
296	0011 0012 0013 0014 0015 0019
297	0011 0012 0014 0015
298	0011 0012 0014 0015
299	0011 0012 0014 0015 0019
300	0011 0012 0014 0015 0019
301	0011 0012 0014 0015 0019
302	0011 0012 0013 0014 0015 0016
303	0011 0012 0013 0014 0015
304	0011 0012 0013 0014 0015
305	0011 0012 0013 0014 0015
306	0011 0012 0013 0014 0015
307	0011 0012 0013 0014 0015 0019
308	0011 0012 0013 0014 0015 0019
309	0011 0012 0013 0014 0015 0019
310	0011 0012 0013 0014 0015 0019
311	0011 0012 0013 0014 0015 0016
312	0011 0012 0013 0014 0015 0016
313	0011 0012 0013 0014 0015
314	0011 0012 0013 0014 0015
315	0011 0012 0013 0014 0015 0019
316	0011 0012 0013 0014 0015
317	0011 0012 0013 0014 0015 0019
318	0011 0012 0013 0014 0015
319	0011 0012 0013 0014 0015 0019
320	0006 0008 0011 0012 0015 0019
321	0011 0012 0014 0015
322	0011 0012 0014 0015

323	0008 0011 0012 0013 0014 0015
324	0011 0012 0014 0015 0019
325	0006 0011 0012 0014 0015
326	0011 0012 0013 0014 0015
327	0011 0012 0013 0014 0015
328	0011 0012 0013 0014 0015
329	0011 0012 0013 0014 0015 0019
330	0011 0012 0013 0014 0015 0016
331	0011 0012 0013 0014 0015 0019
332	0011 0012 0013 0014 0015 0019
333	0008 0011 0012 0013 0015
334	0011 0012 0013 0014 0015
335	0011 0012 0013 0014 0015 0019
336	0011 0012 0013 0014 0015 0016
337	0011 0012 0013 0014 0015
338	0002 0006 0012 0014
339	0011 0012 0013 0014 0015
340	0011 0012 0013 0014 0015
341	0002 0004 0005 0014
342	0006 0008 0009 0012 0016
343	0011 0012 0013 0014 0015
344	0011 0012 0013 0014 0015 0019
345	0011 0012 0014 0015 0019 0020
346	0011 0012 0014 0015 0020
347	0011 0012 0014 0015 0019
348	0011 0012 0014 0015 0019
349	0011 0012 0014 0015 0020
350	0002 0011 0012 0013 0014 0015
351	0011 0012 0013 0014 0015
352	0011 0012 0013 0014 0015

353	0011 0012 0013 0014 0015 0016
354	0011 0012 0013 0014 0015 0019
355	0011 0012 0013 0014 0015 0019
356	0011 0012 0013 0014 0015
357	0011 0012 0013 0014 0015
358	0011 0012 0013 0014 0015 0019
359	0011 0012 0013 0014 0015 0016
360	0011 0012 0013 0014 0015 0019
361	0011 0012 0013 0014 0015
362	0005 0011 0013 0014 0015
363	0011 0012 0013 0014 0015 0016
364	0011 0012 0013 0014 0015 0019
365	0011 0012 0013 0014 0015
366	0011 0012 0013 0014 0015 0018
367	0011 0012 0013 0014 0015 0019
368	0011 0012 0013 0014 0015 0016
369	0011 0012 0014 0015 0019
370	0011 0012 0014 0015 0019
371	0011 0012 0014 0015
372	0011 0012 0014 0015
373	0008 0011 0012 0013 0014 0015
374	0002 0003 0004 0005 0006
375	0002 0003 0004 0005
376	0002 0003 0004 0005 0006
377	0002 0003 0004 0005
378	0002 0003 0004 0005 0009
379	0002 0003 0004 0005 0008
380	0002 0003 0004 0005 0006
381	0002 0003 0004 0005 0006
382	0002 0003 0004 0005 0006

383	0002 0003 0004 0005 0006
384	0002 0003 0004 0005 0006
385	0002 0003 0004 0005 0008
386	0002 0003 0004 0005 0006
387	0002 0003 0004 0005 0006
388	0002 0003 0004 0005 0006
389	0002 0003 0004 0005 0009
390	0002 0003 0004 0005 0009
391	0002 0003 0004 0005
392	0002 0003 0004 0005
393	0002 0003 0004 0005 0006
394	0002 0003 0004 0005 0006
395	0002 0003 0004 0005 0006
396	0002 0003 0004 0005 0006
397	0002 0003 0004 0005 0006
398	0002 0003 0004 0005 0006
399	0002 0003 0004 0005
400	0002 0003 0004 0005 0006
401	0002 0003 0004 0005 0009
402	0002 0003 0004 0005 0009
403	0002 0003 0004 0005 0009
404	0002 0003 0004 0005 0006
405	0002 0003 0004 0005 0008
406	0002 0003 0004 0005 0006
407	0002 0003 0004 0005 0006
408	0002 0003 0004 0005 0006
409	0002 0003 0004 0005 0009
410	0002 0003 0004 0005
411	0002 0003 0004 0005 0006
412	0002 0003 0004 0005 0006

413	0002 0003 0004 0005
414	0002 0003 0004 0005
415	0002 0003 0004 0005 0009
416	0002 0003 0004 0005
417	0002 0003 0004 0005 0009
418	0002 0003 0004 0005 0006
419	0002 0003 0004 0005
420	0002 0003 0004 0005 0006
421	0002 0003 0004 0005 0006
422	0002 0003 0004 0005 0006
423	0002 0003 0004 0005
424	0002 0003 0004 0005 0006
425	0002 0003 0004 0005
426	0002 0003 0004 0005 0008
427	0002 0003 0004 0005
428	0002 0003 0004 0005
429	0002 0003 0004 0005 0006
430	0002 0003 0004 0005 0006
431	0002 0003 0004 0005
432	0002 0003 0004 0005 0009
433	0002 0003 0004 0005
434	0002 0003 0004 0005
435	0002 0003 0004 0005
436	0002 0003 0004 0005
437	0002 0003 0004 0005 0006
438	0002 0003 0004 0005 0009
439	0001 0002 0003 0004 0005
440	0001 0002 0003 0004 0005
441	0002 0003 0004 0005 0006
442	0002 0003 0004 0005 0009

443	0002 0003 0004 0005
444	0001 0002 0003 0004 0005
445	0002 0003 0004 0005
446	0002 0003 0004 0005
447	0002 0003 0004 0005 0009
448	0002 0003 0004 0005 0006
449	0002 0003 0004 0005 0006
450	0002 0003 0004 0005
451	0002 0003 0004 0005
452	0002 0003 0004 0005
453	0002 0003 0004 0005
454	0002 0003 0004 0005 0006
455	0002 0003 0004 0005 0006
456	0002 0003 0004 0005 0009
457	0002 0003 0004 0005 0006
458	0002 0003 0004 0005 0006
459	0002 0003 0004 0005 0006
460	0002 0003 0004 0005 0008
461	0002 0003 0004 0005 0009
462	0002 0003 0004 0005 0009
463	0002 0003 0004 0005 0009
464	0002 0003 0004 0005
465	0002 0003 0004 0005
466	0002 0003 0004 0005 0006
467	0002 0003 0004 0005 0006
468	0002 0003 0004 0005
469	0002 0003 0004 0005 0006
470	0002 0003 0004 0005 0006
471	0002 0003 0004 0005 0009
472	0002 0003 0004 0005 0006

473	0002 0003 0004 0005 0006
474	0002 0003 0004 0005
475	0002 0003 0004 0005 0009
476	0002 0003 0004 0005 0009
477	0002 0003 0004 0005
478	0002 0003 0004 0005 0008
479	0002 0003 0004 0005 0006
480	0002 0003 0004 0005 0006
481	0002 0003 0004 0005
482	0002 0003 0004 0005
483	0002 0003 0004 0005 0006
484	0002 0003 0004 0005
485	0002 0003 0004 0005
486	0002 0003 0004 0005
487	0002 0003 0004 0005 0006
488	0002 0003 0004 0005 0006
489	0002 0003 0004 0005
490	0002 0003 0004 0005 0006
491	0002 0003 0004 0005
492	0002 0003 0004 0005
493	0002 0003 0004 0005
494	0002 0003 0004 0005
495	0002 0003 0004 0005
496	0002 0003 0004 0005
497	0002 0003 0004 0005 0009
498	0002 0003 0004 0005 0006
499	0002 0003 0004 0005
500	0002 0003 0004 0005 0008
501	0002 0003 0004 0005
502	0002 0003 0004 0005

503	0002 0003 0004 0005 0009
504	0002 0003 0004 0005
505	0002 0003 0004 0005 0009
506	0002 0003 0004 0005
507	0002 0003 0004 0005 0006
508	0002 0003 0004 0005
509	0002 0003 0004 0005
510	0001 0002 0003 0004 0005
511	0002 0003 0004 0005 0009
512	0002 0003 0004 0005
513	0002 0003 0004 0005
514	0002 0003 0004 0005 0006
515	0002 0003 0004 0005 0006
516	0002 0003 0004 0005
517	0002 0003 0004 0005 0006
518	0002 0003 0004 0005 0006
519	0002 0003 0004 0005